1. Introduction to Microprocessor

Definition:

- *"The microprocessor is a multipurpose, clock driven, register based, digital-integrated circuit which accepts binary data as input, processes it according to instructions stored in its memory, and provides results as output."*
- "Microprocessor is a computer Central Processing Unit (CPU) on a single chip that contains millions of transistors connected by wires."

Introduction:

- A microprocessor is designed to perform arithmetic and logic operations that make use of small number-holding areas called registers.
- Typical microprocessor operations include adding, subtracting, comparing two numbers, and fetching numbers from one area to another.

2. Components of Microprocessor

- Microprocessor is capable of performing various computing functions and making decisions to change the sequence of program execution.
- The microprocessor can be divided into three segments as shown in the figure, Arithmetic/logic unit (ALU), register array, and control unit.
- These three segment is responsible for all processing done in a computer

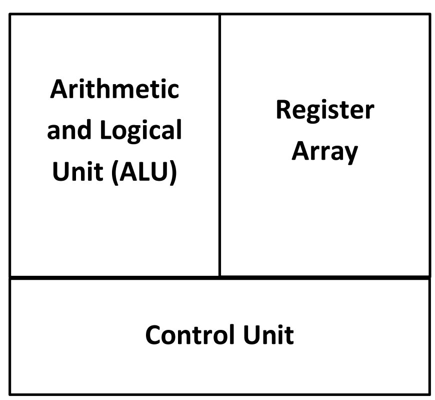


Figure: Components of Microprocessor

Arithmetic and logic unit (ALU)

- It is the unit of microprocessor where various computing functions are performed on the data.
- It performs arithmetic operations such as addition, subtraction, and logical operations such as OR,AND, and Exclusive-OR.
- It is also known as the brain of the computer system.

Register array

- It is the part of the register in microprocessor which consists of various registers identified by letters such as B, C, D, E, H, and L.
- Registers are the small additional memory location which are used to store and transfer data and programs that are currently being executed.

Control unit

- The control unit provides the necessary timing and control signals to all the operations in the microcomputer.
- It controls and executes the flow of data between the microprocessor, memory and peripherals.
- The control bus is bidirectional and assists the CPU in synchronizing control signals to internal devices and external components.
- This signal permits the CPU to receive or transmit data from main memory.

3. System bus (data, address and control bus).

- This network of wires or electronic pathways is called the 'Bus'.
- A system bus is a single computer bus that connects the major components of a computer system.
- It combines the functions of a data bus to carry information, an address bus to determine where it should be sent, and a control bus to determine its operation.
- The technique was developed to reduce costs and improve modularity.

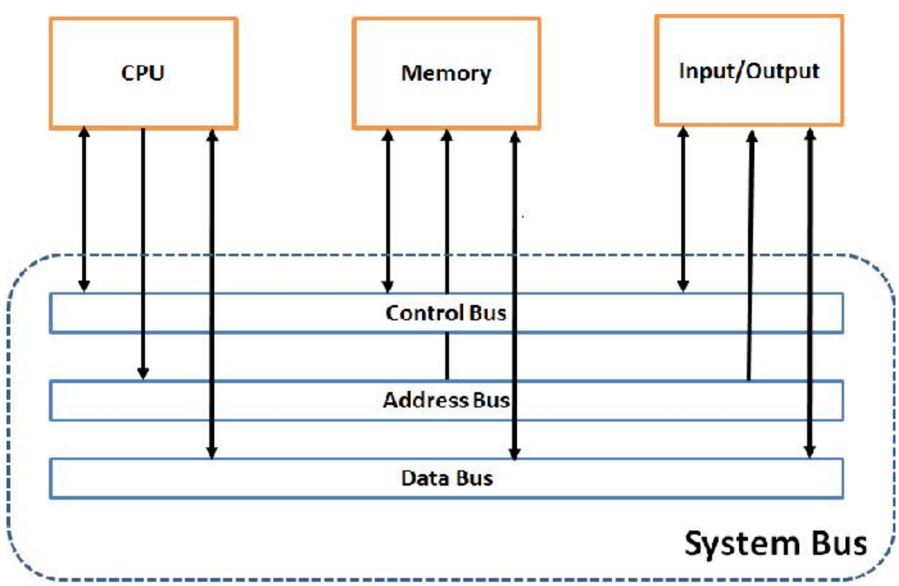


Figure: System Bus

Address Bus

- It is a group of wires or lines that are used to transfer the addresses of Memory or I/O devices.
- It is unidirectional.
- The width of the address bus corresponds to the maximum addressing capacity of the bus, or the largest address within memory that the bus can work with.
- The addresses are transferred in binary format, with each line of the address bus carrying a single binary digit.
- Therefore the maximum address capacity is equal to two to the power of the number of lines present (2^lines).

Data Bus

- It is used to transfer data within Microprocessor and Memory/Input or Output devices.
- It is bidirectional as Microprocessor requires to send or receive data.
- Each wire is used for the transfer of signals corresponding to a single bit of binary data.
- As such, a greater width allows greater amounts of data to be transferred at the same time.

Control Bus

- Microprocessor uses control bus to process data, i.e. what to do with the selected memory location.
- Some control signals are Read, Write and Opcode fetch etc.
- Various operations are performed by microprocessor with the help of control bus.
- This is a dedicated bus, because all timing signals are generated according to control signal.

4. Microprocessor systems with bus organization

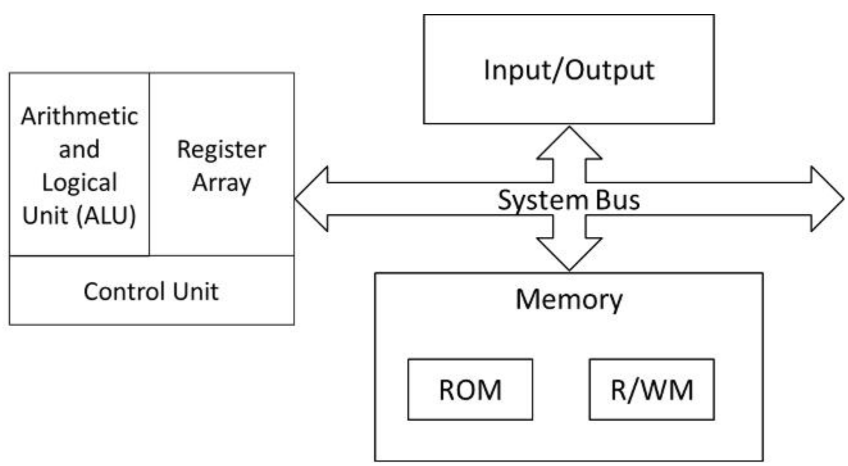


Figure: Microprocessor systems with bus organization

- To design any meaningful application microprocessor requires support of other auxiliary devices.
- In most simplified form a microprocessor based system consist of a microprocessor, I/O (input/output) devices and memory.
- These components are interfaced (connected) with microprocessor over a common communication path called system bus. Typical structure of a microprocessor based system is shown in Figure.
- Here, microprocessor is master of the system and responsible for executing the program and coordinating with connected peripherals as required.
- Memory is responsible for storing program as well as data. System generally consists of two types of memories ROM (Read only and non-volatile) and RAM (Read/Write and volatile).
- I/O devices are used to communicate with the environment. Keyboard can be example of input devices and LED, LCD or monitor can be example of output device.
- Depending on the application level of sophistication varies in a microprocessor based systems. For example: washing machine, computer.

1. Explain Classification of Memory

Ans.

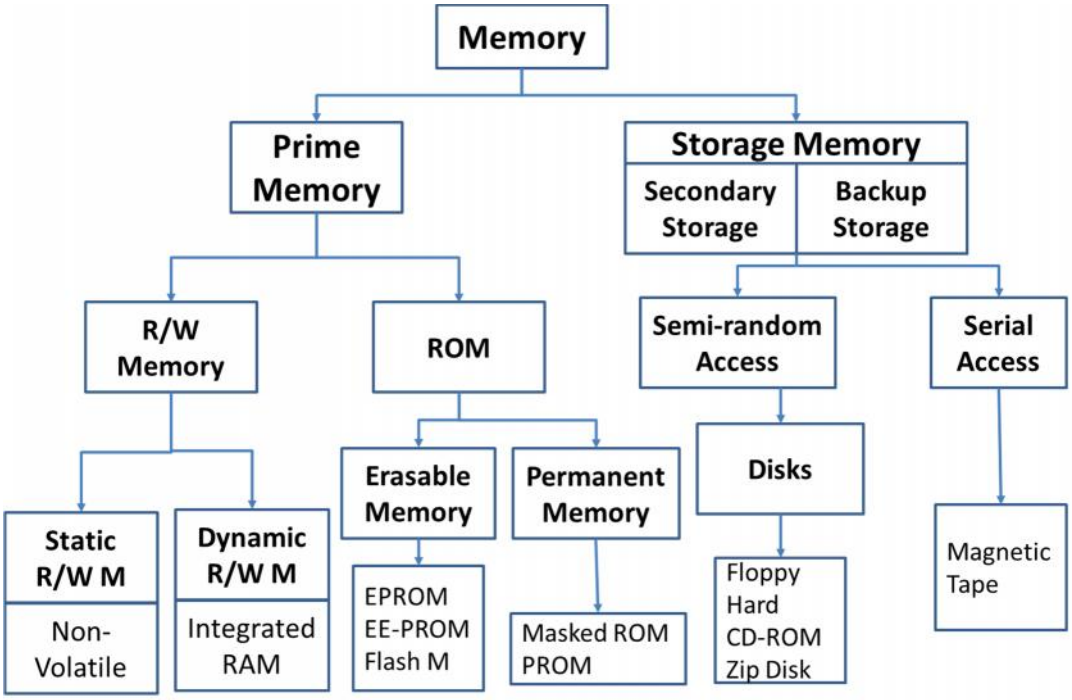


Figure: Classification of Memory

ROM (Read Only Memory):

The first classification of memory is ROM. The data in this memory can only be read, no writing is allowed. It is used to store permanent programs. It is a nonvolatile type of memory.

The classification of ROM memory is as follows:

1. **Masked ROM**: the program or data are permanently installed at the time of manufacturing as per requirement. The data cannot be altered. The process of permanent recording is expensive but economic for large quantities.

2. **PROM (Programmable Read Only Memory):** The basic function is same as that of masked ROM. but in PROM, we have fuse links. Depending upon the bit pattern, the fuse can be burnt or kept intact. This job is performed by PROM programmer.
   To do this, it uses high current pulse between two lines. Because of high current, the fuse will get burnt; effectively making two lines open. Once a PROM is programmed we cannot change connections, only a facility provided over masked ROM is, the user can load his program in it. The disadvantage is a chance of re-growing of the fuse and changes the programmed data because of aging.

3. **EPROM (Erasable Programmable Read Only Memory):** the EPROM is programmable by the user. It uses MOS circuitry to store data. They store 1's and 0's in form of charge. The information stored can be erased by exposing the memory to ultraviolet light which erases the data stored in all memory locations. For ultraviolet light, a quartz window is provided which is covered during normal operation. Upon erasing it can be reprogrammed by using EPROM programmer. This type of memory is used in a project developed and for experiment use. The advantage is it can be programmed erased and reprogrammed. The disadvantage is all the data get erased even if you want to change single data bit.

4. **EEPROM:** EEPROM stands for electrically erasable programmable read only memory. This is similar to EPROM except that the erasing is done by electrical signals instead of ultraviolet light. The main advantage is the memory location can be selectively erased and reprogrammed. But the manufacturing process is complex and expensive so do not commonly used.

R/W Memory (Read/Write Memory):

The RAM is also called as read/write memory. The RAM is a volatile type of memory. It allows the programmer to read or write data. If the user wants to check the execution of any program, user feeds the program in RAM memory and executes it. The result of execution is then checked by either reading memory location contents or by register contents.

Following is the classification of RAM memory.
It is available in two types:

1. **SRAM (Static RAM):** SRAM consists of the flip-flop; using either transistor or MOS. for each bit we require one flip-flop. Bit status will remain as it is; unless and until you perform next write operation or power supply is switched off.

   Advantages of SRAM:
   - Fast memory (less access time)
   - Refreshing circuit is not required.

   Disadvantages of SRAM:
   - Low package density
   - Costly

2. **DRAM (Dynamic RAM):** In this type of memory a data is stored in form of charge in capacitors. When data is 1, the capacitor will be charged and if data is 0, the capacitor will not be charged. Because of capacitor leakage currents, the data will not be held by these cells. So the DRAMs require refreshing of memory cells. It is a process in which same data is read and written after a fixed interval.

Advantages of DRAM:

- High package density
- Low cost

Disadvantages of DRAM:

- Required refreshing circuit to maintain or refresh charge on the capacitor, every after few milliseconds.

Secondary Memory

- **Magnetic Disk**: The Magnetic Disk is Flat, circular platter with metallic coating that is rotated beneath read/write heads. It is a Random access device; read/write head can be moved to any location on the platter

- **Floppy Disk**: These are small removable disks that are plastic coated with magnetic recording material. Floppy disks are typically 3.5" in size (diameter) and can hold 1.44 MB of data. This portable storage device is a rewritable media and can be reused a number of times. Floppy disks are commonly used to move files between different computers. The main disadvantage of floppy disks is that they can be damaged easily and, therefore, are not very reliable. The following figure shows an example of the floppy disk. Figure 3 shows a picture of the floppy disk.

- **Hard Disk**: Another form of auxiliary storage is a hard disk. A hard disk consists of one or more rigid metal plates coated with a metal oxide material that allows data to be magnetically recorded on the surface of the platters. The hard disk platters spin at 5 a high rate of speed, typically 5400 to 7200 revolutions per minute (RPM).Storage capacities of hard disks for personal computers range from 10 GB to 120 GB (one billion bytes are called a gigabyte).

- **Optical Disks:** Optical Mass Storage Devices Store bit values as variations in light reflection. They have higher area density & longer data life than magnetic storage. They are also standardized and relatively inexpensive. Their Uses: read-only storage with low performance requirements, applications with high capacity requirements & where portability in a standardized format is needed.

Types of Optical Disk
1. CD-ROM (read only)
2. CD-R: (record) to a CD
3. CD-RW: can write and erase CD to reuse it (re-writable)
4. DVD(Digital Video Disk)

2. Explain I/O devices and their Interfacing

Ans. Input / Output (I/O)

- MPU communicates with outside word through I/O device.
- There are 2 different methods by which MPU identifies and communicates With I/O devices these methods are:
  1- Direct I/O (Peripheral)
  2- Memory-Mapped I/O

The methods differ in terms of the

- No. of address lines used in identifying an I/O device.
- Type of control lines used to enable the device.
- Instructions used for data transfer.

Direct I/O (Peripheral):-

- This method uses two instructions (IN & OUT) for data transfer.
- MPU uses 8 address lines to send the address of I/O device (can identify 256 input devices & 256 output devices).
- The (I/P & O/P devices) can be differentiated by control signals I/O Read (IOR) and I/O Write (IOW).
- The steps in communicating with an I/O device are similar to those in communicating with memory and can be summarized as follows:
  1- The MPU places an 8-bit device address on address bus then decoded.
  2- The MPU sends a control signal (IOR or IOW) to enable the I/O device.
  3- Data are placed on the data bus for transfer.

Memory-Mapped I/O:-

- The MPU uses 16 address lines to identify an I/O device.
- This is similar to communicating with a memory location.
- Use the same control signals (MEMR or MEMW) and instructions as those of memory.
- The MPU views these I/O devices as if they were memory locations.
- There are no special I/O instructions.
- It can identify 64k address shared between memory & I/O devices.

1. Write down main features of 8085 microprocessor.

- It is an 8 bit microprocessor.
- It is manufactured with N-MOS technology.
- It has 16-bit address bus and hence can address up to 216 = 65536 bytes (64KB) memory locations through A0-A15
- The first 8 lines of address bus and 8 lines of data bus are multiplexed AD0 – AD7
- Data bus is a group of 8 lines D0 – D7
- It supports external interrupt request. .
- A 16 bit program counters (PC)
- A 16 bit stack pointer (SP)
- Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
- It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.
- It is enclosed with 40 pins DIP (Dual in line package).

2. Explain 8085 microprocessor architecture.

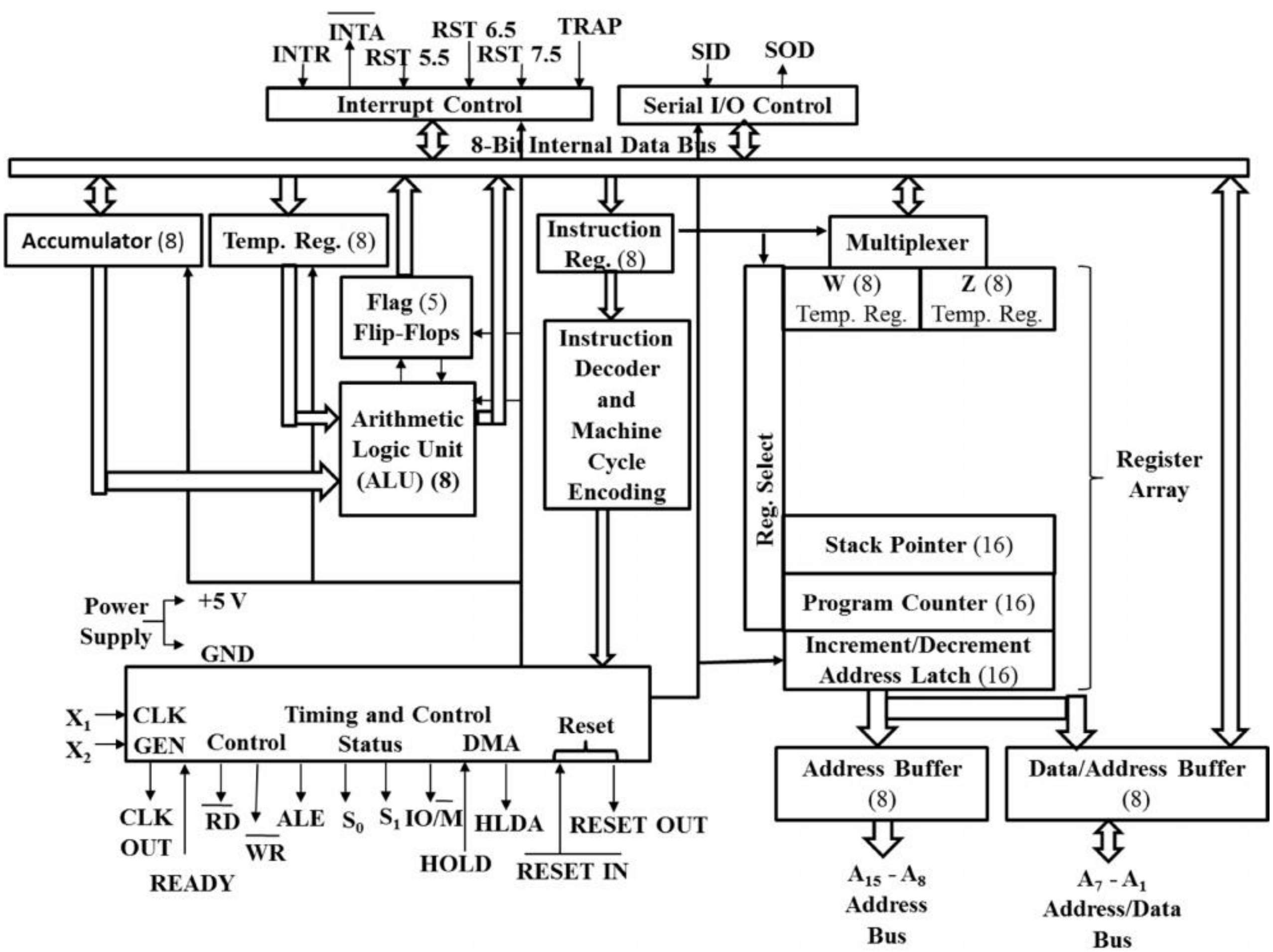


Figure: 8085 microprocessor architecture.

- The architecture of microprocessor 8085 can be divided into seven parts as follows:

Register Unit:

General Purpose Data Register

- 8085 has six general purpose data registers to store 8-bit data.
- These registers are named as B, C, D, E, H and L as shown in fig. 1.
- The user can use these registers to store or copy a data temporarily during the execution of a program by using data transfer instructions.
- These registers are of 8 bits but whenever the microprocessor has to handle 16-bit data, these registers can be combined as register pairs – BC, DE and HL.
- There are two internal registers – W and X. These registers are only for internal operation like execution of CALL and XCHG instructions and not available to the user.

Program Counter (PC)

- 16-bit register deals with sequencing the execution of instructions.
- This register is a memory pointer.
- Memory locations have 16-bit addresses which are why this is a 16-bit register.
- The microprocessor uses this register to sequence the execution of the instructions.
- The function of the program counter is to point to the memory address from which the next byte is to be fetched.
- When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.

Stack Pointer (SP)

- SP is also a 16-bit register used as a memory pointer.
- It points to a memory location in R/W memory, called the stack.
- The beginning of the stack is defined by loading 16-bit address in the stack pointer.

MUX/DEMUX unit

- This unit is used to select a register out of all the available registers.
- This unit behaves as a MUX when data is going from the register to the internal data bus.
- It behaves as a DEMUX when data is coming to a register from the internal data bus of the microprocessor.
- The register select will behave as the function selection lines of the MUX/DEMUX.

Address Buffer Register & Data/Address Buffer Register

- These registers hold the address/data, received from PC/internal data bus and then load the external address and data buses.
- These registers actually behave as the buffer stage between the microprocessor and external system buses.

Control Unit:

- The control unit generates signals within microprocessor to carry out the instruction, which has been decoded.
- In reality it causes connections between blocks of the microprocessor to be opened or closed, so that the data goes where it is required and the ALU operations occur.
- The control unit itself consists of three parts; the instruction registers (IR), instruction decoder and machine cycle encoder and timing and control unit.

Instruction Register

- This register holds the machine code of the instruction.
- When microprocessor executes a program it reads the opcode from the memory, this opcode is stored in the instruction register.

Instruction Decoder & Machine Cycle Encoder

- The IR sends the machine code to this unit.
- This unit, as its name suggests, decodes the opcode and finds out what is to be done in response of the coming opcode and how many machine cycles are required to execute this instruction.

Timing & Control unit

- The control unit generates signals within microprocessor to carry out the instruction, which has been decoded.
- In reality, it causes certain connections between blocks of the microprocessor to be opened or closed, so that the data goes where it is required and the ALU operations occur.

Arithmetic & Logical Unit:

- The ALU performs the actual numerical and logical operation such as 'add', 'subtract', 'AND', 'OR', etc.
- ALU uses data from memory and from accumulator to perform the arithmetic operations and always stores the result of the operation in accumulator.
- ALU consists of accumulator, flag register and temporary register.

Accumulator

- The accumulator is an 8-bit register that is a part of ALU.
- This register is used to store 8-bit data and perform arithmetical and logical operations.
- The result of an operation is stored in the accumulator.
- It is also identified as register A.

Flags register

- Flag register includes five flip-flops, which are set or reset after an operation according to the data conditions of the result in the accumulator and other registers.
- They are called zero (Z), carry (CY), sign (S), parity (P) and auxiliary carry (AC) flags; their bit positions in the flag register are shown in fig.
- The microprocessor uses these flags to set and test data conditions.

Interrupt Control

- The interrupt control unit has 5 interrupt inputs TRAP,RST 7.5, RST 6.5, RST 5.5 & INTR and one acknowledge signal INTA.

- It controls the interrupt activity of 8085 microprocessor.

Serial IO control

- 8085 serial IO control provides two lines, SOD and SID for serial communication.
- The serial output data (SOD) line is used to send data serially and serial input data line (SID) is used to receive data serially.

3. Explain Flags Registers in 8085

- Flag register includes five flip-flops, which are set or reset after an operation according to the data conditions of the result in the accumulator and other registers.
- They are called zero (Z), carry (CY), sign (S), parity (P) and auxiliary carry (AC) flags; their bit positions in the flag register are shown in fig.
- The microprocessor uses these flags to set and test data conditions.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

Figure: Flags registers in 8085.

- The flags are stored in the 8-bit register so that the programmer can examine these flags by accessing the register through an instruction.
- These flags have critical importance in the decision-making process of the microprocessor.
- The conditions (set or reset) of the flags are tested through the software instructions.
- For instance, JC (jump on carry) is implemented to change the sequence of a program when CY flag is set.

Z (Zero) Flag:

- This flag indicates whether the result of mathematical or logical operation is zero or not.
- If the result of the current operation is zero, then this flag will be set, otherwise reset.

CY (Carry) Flag:

- This flag indicates, whether, during an addition or subtraction operation, carry or borrow is generated or not, if generated then this flag bit will be set.

AC (Auxiliary Carry) Flag:

- It shows carry propagation from D3 position to D4 position.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	1	0	0
0	0	1	0	1	0	1	1
1	0	1	1	0	1	1	1

Figure: Auxiliary Carry.

- As shown in the fig., a carry is generated from D3 bit position and propagates to the D4 position. This carry is called auxiliary carry.

S (Sign) Flag:

- Sign flag indicates whether the result of a mathematical operation is negative or positive.
- If the result is positive, then this flag will reset and if the result is negative this flag will be set.
- This bit, in fact, is a replica of the D7 bit.

P (Parity) Flag:

- Parity is the number of 1’s in a number.
- If the number of 1’s in a number is even then that number is known as even parity number.
- If the number of 1’s in a number is odd then that number is known as an odd parity number.
- This flag indicates whether the current result is of even parity (set) or of odd parity (reset).

4. Explain 8085 pin diagram.

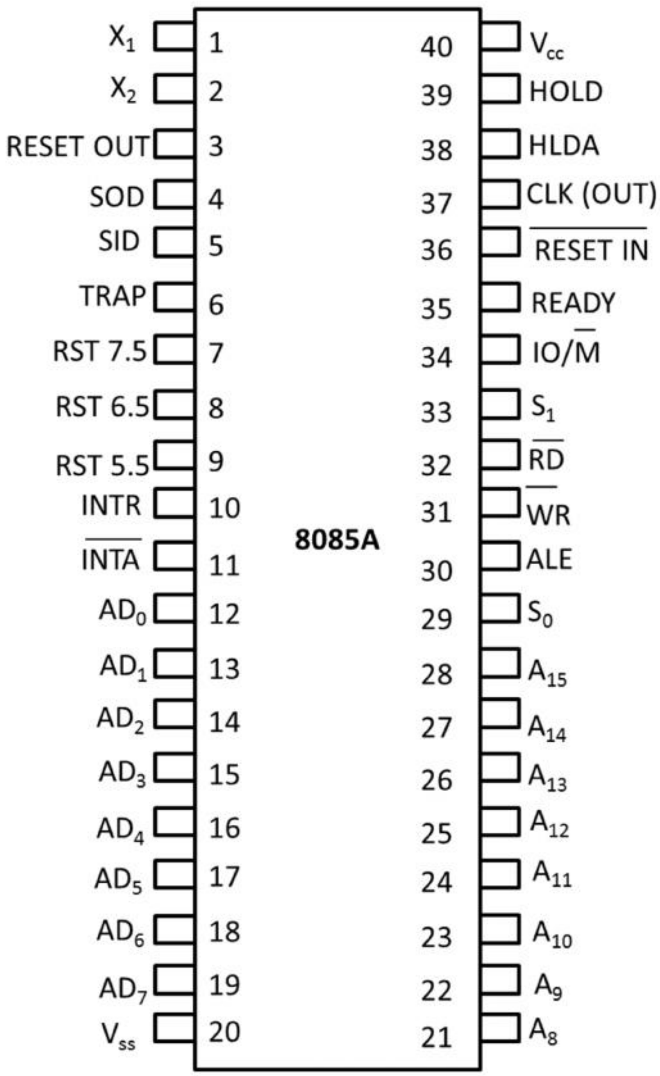


Figure: 8085 pin diagram.

- All signals can be classified into six groups:
  1. Address Bus
  2. Data Bus
  3. Control & Status Signals
  4. Power Supply & Frequency signals
  5. Externally initiated signals
  6. Serial I/O Ports

1) Address Bus (pin 12 to 28)

- 16 signal lines are used as address bus.
- However these lines are split into two segments: A_{15} - A_8 and AD_7 - AD_0
- A_{15} - A_8 are unidirectional and are used to carry high-order address of 16-bit address.
- AD_7 - AD_0 are used for dual purpose.

2) Data Bus/ Multiplexed Address (pin 12 to 19)

- Signal lines AD7-AD0 are bidirectional and serve dual purpose.
- They are used as low-order address bus as well as data bus.
- The low order address bus can be separate from these signals by using a latch.

3) Control & Status Signals

- To identify nature of operation
- Two Control Signals
  1) RD' (Read-pin 32)
     - ✓ This is a read control signal (active low)
     - ✓ This signal indicates that the selected I/O or Memory device is to be read & data are available on data bus.
  2) WR' (Write-pin 31)
     - ✓ This is a write control signal (active low)
     - ✓ This signal indicates that the selected I/O or Memory device is to be write.
- Three Status Signals
  1) S_1 (pin 33)
  2) S_0 (pin 29)
     - ✓ S_1 and S_0 status signals can identify various operations, but they are rarely used in small systems.

S_1	S_0	Mode
0	0	HLT
0	1	WRITE
1	0	READ
1	1	OPCODE FETCH

  3) IO/M' (pin 34)
     - ✓ This is a status signal used to differentiate I/O and memory operation
     - ✓ When it is high, it indicates an I/O operation
     - ✓ When it is low, it indicates a memory operation
     - ✓ This signal is combined with RD' and WR' to generate I/O & memory control signals
- To indicate beginning of operation
  - o One Special Signal called ALE (Address Latch Enable-Pin 30)
  - o This is positive going pulse generated every time the 8085 begins an operation (machine cycle)
  - o It indicates that the bits on AD7-AD0 are address bits
  - o This signal is used primarily to latch the low-address from multiplexed bus & generate a separate set of address lines A7-A0.

4) Power Supply & Frequency Signal

- V_{cc}→Pin no. 40, +5V Supply
- V_{ss} → Pin no.20, Ground Reference
- X1, X2 → Pin no.1 & 2, Crystal Oscillator is connected at these two pins. The frequency is internally divided by two;
  - Therefore, to operate a system at 3MHz, the crystal should have a frequency of 6MHz.
- CLK (OUT) → Clock output. Pin No.37: This signal can be used as the system clock for other devices.

5) Externally Initiated Signals including Interrupts

- INTR (Input) → Interrupt Request. It is used as general purpose interrupt
- INTA' (Output) → Interrupt Acknowledge. It is used to acknowledge an interrupt.
- RST7.5, RST6.5, RST5.5 (Input) → Restart Interrupts.
  - These are vector interrupts that transfer the program control to specific memory locations.
  - They have higher priorities than INTR interrupt.
  - Among these 3 interrupts, the priority order is RST7.5, RST6.5, RST5.5
- TRAP (Input) → This is a non maskable interrupt & has the highest priority.
- HOLD (Input) → This signal indicates that a peripheral such as DMA Controller is requesting the use of address & data buses
- HLDA (Output) → Hold Acknowledge. This signal acknowledges the HOLD request
- READY (Input) → This signal is used to delay the microprocessor read or write cycles until as low-responding peripheral is ready to send or accept data. When the signal goes low, the microprocessor waits for an integral no. of clock cycles until it goes high.
- RESET IN' (Input) → When the signal on this pin goes low, the Program Counter is set to zero, the buses are tri-stated & microprocessor is reset.
- RESET OUT (Output) → This signal indicates that microprocessor is being reset. The signal can be used to reset other devices.

6) Serial I/O Ports

- Two pins for serial transmission
  1) SID (Serial Input Data-pin 5)
  2) SOD (Serial Output Data-pin 4)
- In serial transmission, data bits are sent over a single line, one bit at a time.

5. Explain Instruction Cycle

- Instruction Cycle is defined as time required to complete execution of an instruction.
- 8085 instruction cycle consists of 1 to 6 Machine Cycles or 1 to 6 operations.

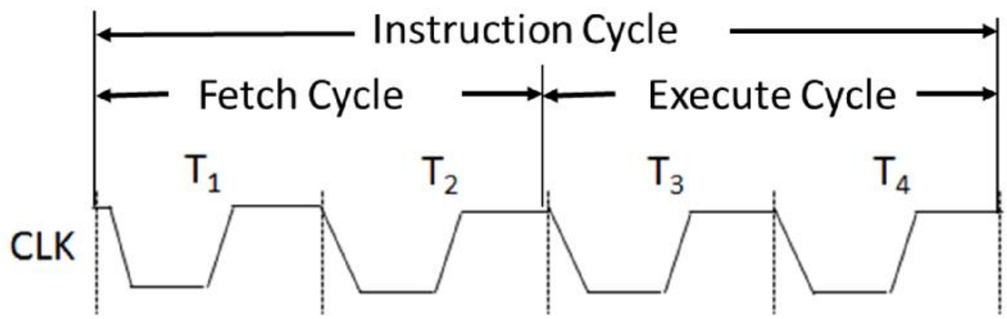


Figure: Instruction Cycle.

6. Explain Machine Cycle

- Machine Cycle is defined as time required by the microprocessor to complete operation of accessing memory device or I/O device.
- This cycle may consist 3 to 6 T-states.
- The basic microprocessor operation such as reading a byte from I/O port or writing a byte to memory is called as machine cycle.

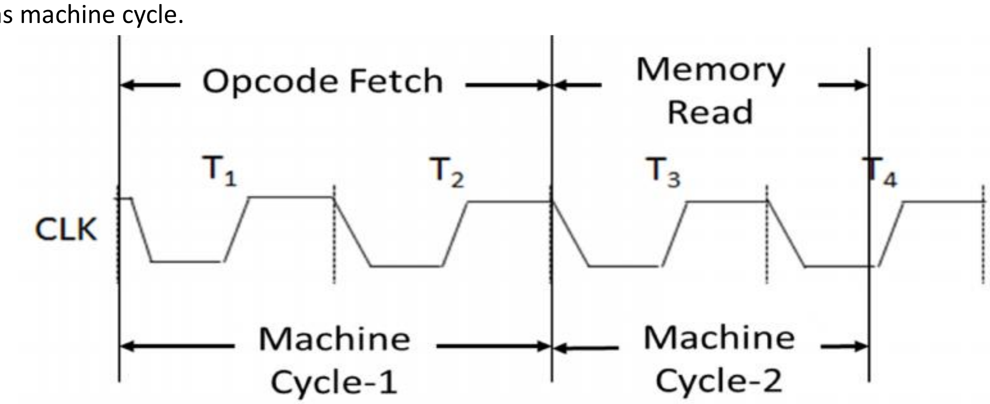


Figure: Machine Cycle.

7. Explain T-States

- T-States are defined as one subdivision of operation performed in one clock period.
- These sub divisions are internal states synchronized with system clock & each T-state is precisely equal to one clock period.

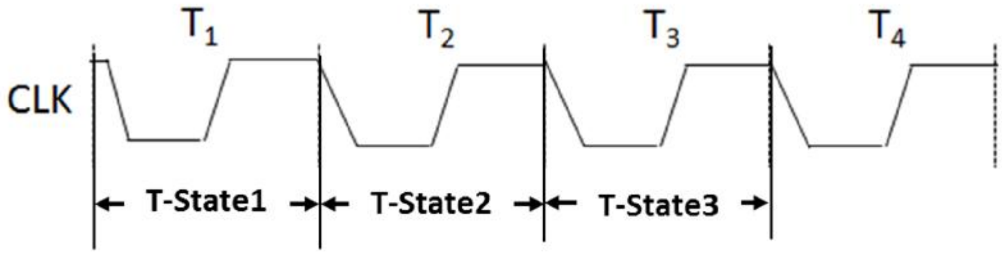


Figure: T-States.

8. Compare Instruction Cycle, Machine Cycle and T-States

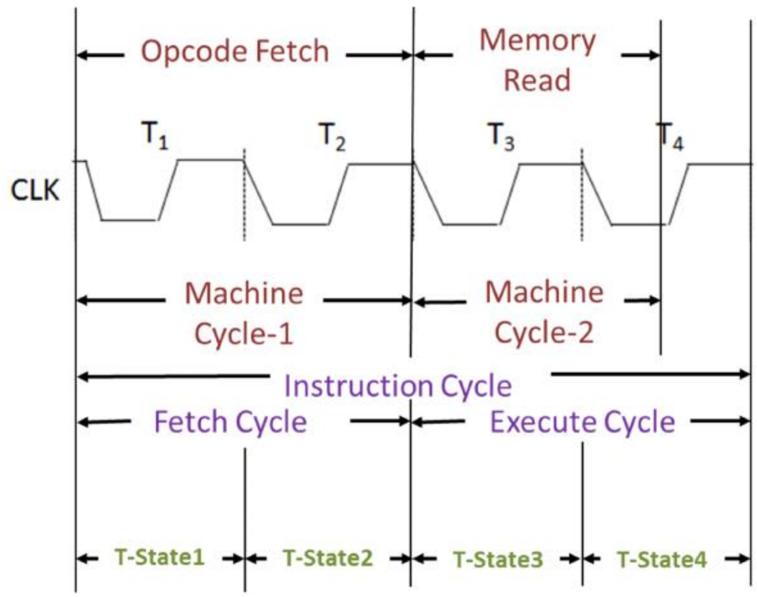


Figure: Comparison between Instruction Cycle, Machine Cycle and T-States.

- Instruction Cycle: Time required to complete execution of an instruction.
- Machine Cycle: Time required by the microprocessor to complete an operation.
- T-States: One subdivision of operation performed in one clock period.

9. Explain 8085 Programming Model

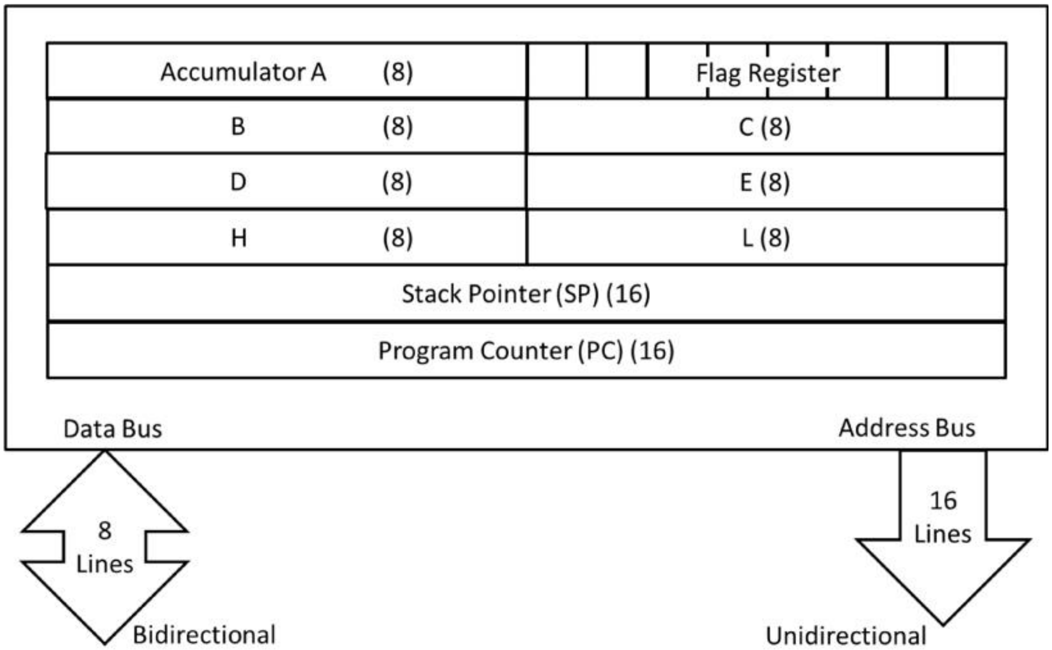


Figure: 8085 Programming Model.

Registers

- 6 general purpose registers to store 8-bit data B, C, D, E, H & L.
- Can be combined as register pairs – BC, DE, and HL to perform 16-bit operations.
- Used to store or copy data using data copy instructions.

Accumulator

- 8 - bit register, identified as A
- Part of ALU
- Used to store 8-bit data to perform arithmetic & logical operations.
- Result of operation is stored in it.

Flag Register

- ALU has 5 Flag Register that set/reset after an operation according to data conditions of the result in accumulator & other registers.
- Helpful in decision making process of Microprocessor
- Conditions are tested through software instructions
- For e.g.
- JC (Jump on Carry) is implemented to change the sequence of program when CY is set.

Program Counter

- 16-bit registers used to hold memory addresses.
- Size is 16-bits because memory addresses are of 16-bits.

- Microprocessor uses PC register to sequence the execution of instructions.
- Its function is to point to memory address from which next byte is to be fetched.
- When a byte is being fetched, PC is incremented by 1 to point to next memory location.

Stack Pointer

- Used as memory pointer
- Points to the memory location in R/W memory, called Stack.
- Beginning of stack is defined by loading a 16-bit address in the stack pointer.

10. Explain Bus Organization of 8085

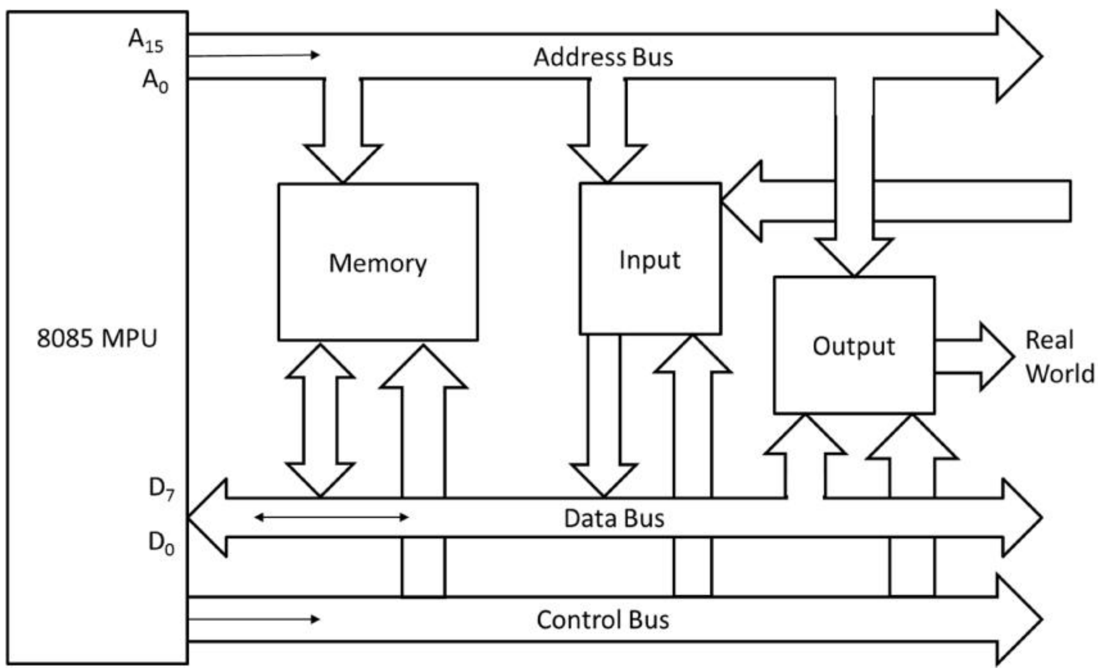


Figure: Bus Organization of 8085.

Address Bus

- Group of 16 lines generally identified as A0 to A15.
- It is unidirectional i.e. bits flow from microprocessor to peripheral devices.
- 16 address lines are capable of addressing 65536 memory locations.
- So, 8085 has 64K memory locations.

Data Bus

- Group of 8 lines identified as D0 to D7.
- They are bidirectional i.e. data flow in both directions between microprocessor, memory & peripheral.
- 8 data lines enable microprocessor to manipulate data ranging from 00H to FFH (28=256 numbers).
- Largest number appear on data bus is 1111 1111 => (255)10.
- As Data bus is of 8-bit, 8085 is known as 8-bit Microprocessor.

Control Bus

- It comprises of various single lines that carry synchronization, timing & control signals.

- These signals are used to identify a device type with which MPU intends to communicate.

11. Explain Demultiplexing AD0-AD7

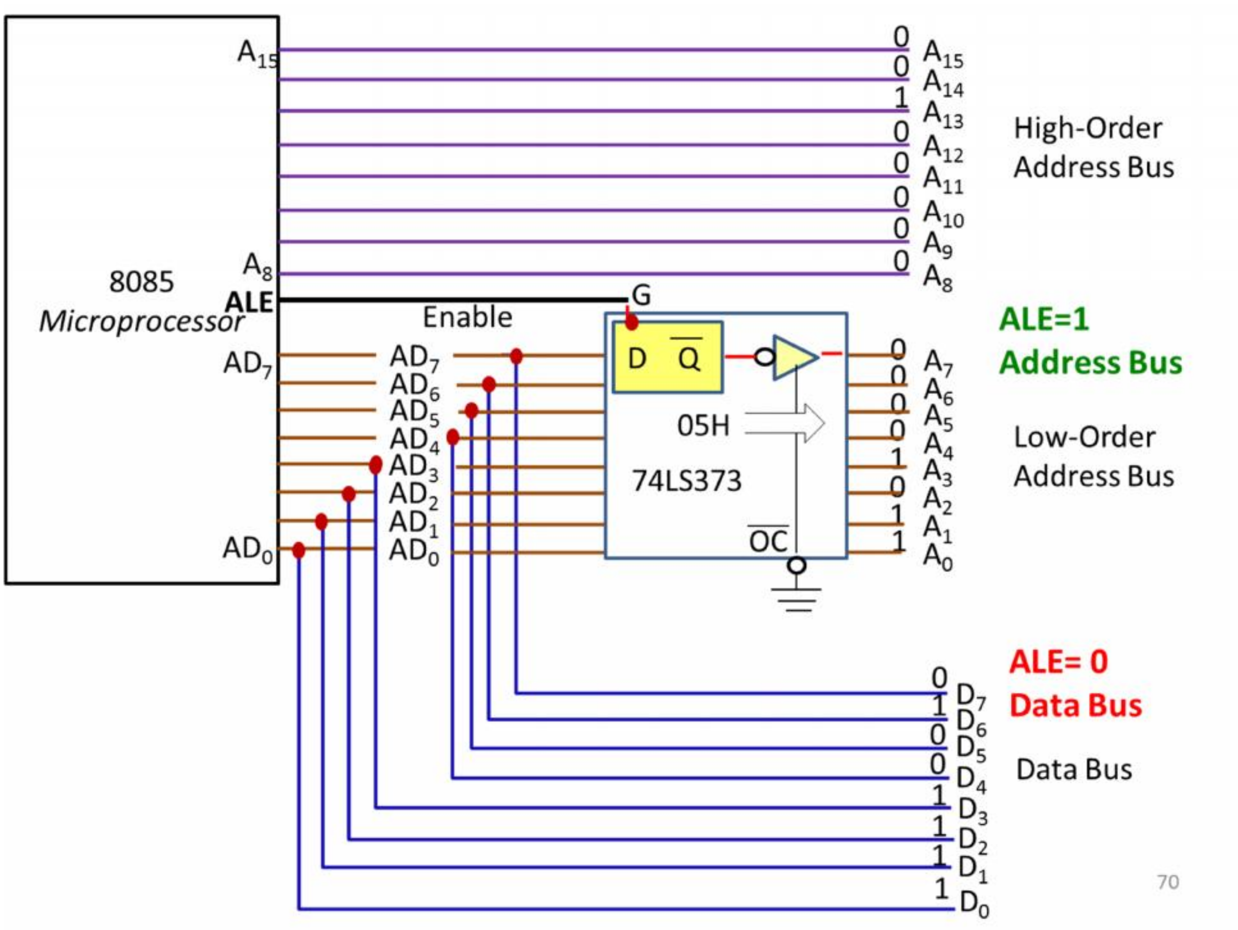


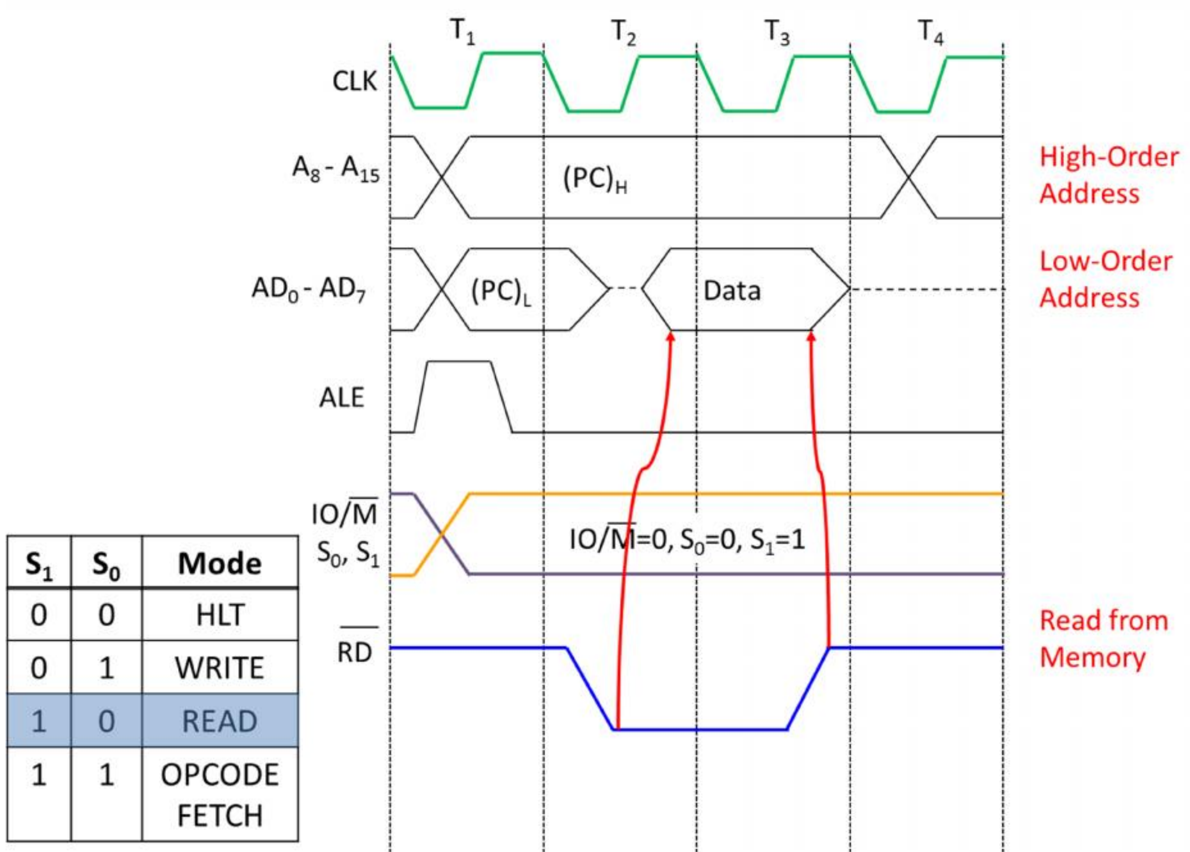
Figure: Demultiplexing AD0-AD7.

- The higher-order bus remains on the bus for three clock periods. However, the low-order address is lost after the first clock period.
- This address need to be latched and used for identifying the memory address. If the bus AD7-AD0 is used to identify the memory location (2005H), the address will change to 204FH after the first clock period.
- Figure shows a schematic that uses a latch and the ALE signal to demultiplex the bus.
- The bus AD7-AD0 is connected as the input to the latch.
- The ALE signal is connected to the Enable pin of the latch, and the output control signal of the latch is grounded.
- Figure shows that the ALE goes high during T1. And during T1 address of lower-order address bus is store into the latch.

12. Explain Memory Interfacing

- When we are executing any instruction, we need the microprocessor to access the memory for reading instruction codes and the data stored in the memory.
- For this, both the memory and the microprocessor requires some signals to read/write to/from registers.
- The interfacing circuit therefore should be designed in such a way that it matches the memory signal requirements with the signals of the microprocessor.

Memory Read Cycle



S_1	S_0	Mode
0	0	HLT
0	1	WRITE
1	0	READ
1	1	OPCODE FETCH

Figure: Memory Read Cycle.

- It is used to fetch one byte from the memory.
- It requires 3 T-States.
- It can be used to fetch operand or data from the memory.
- During T1, A8-A15 contains higher byte of address. At the same time ALE is high. Therefore Lower byte of address A0-A7 is selected from AD0-AD7.
- Since it is memory ready operation, IO/M (bar) goes low.
- During T2 ALE goes low, RD (bar) goes low. Address is removed from AD0-AD7 and data D0-D7 appears on AD0-AD7.
- During T3, Data remains on AD0-AD7 till RD (bar) is at low signal.

Memory Write Cycle

S_1	S_0	Mode
0	0	HLT
0	1	WRITE
1	0	READ
1	1	OPCODE FETCH

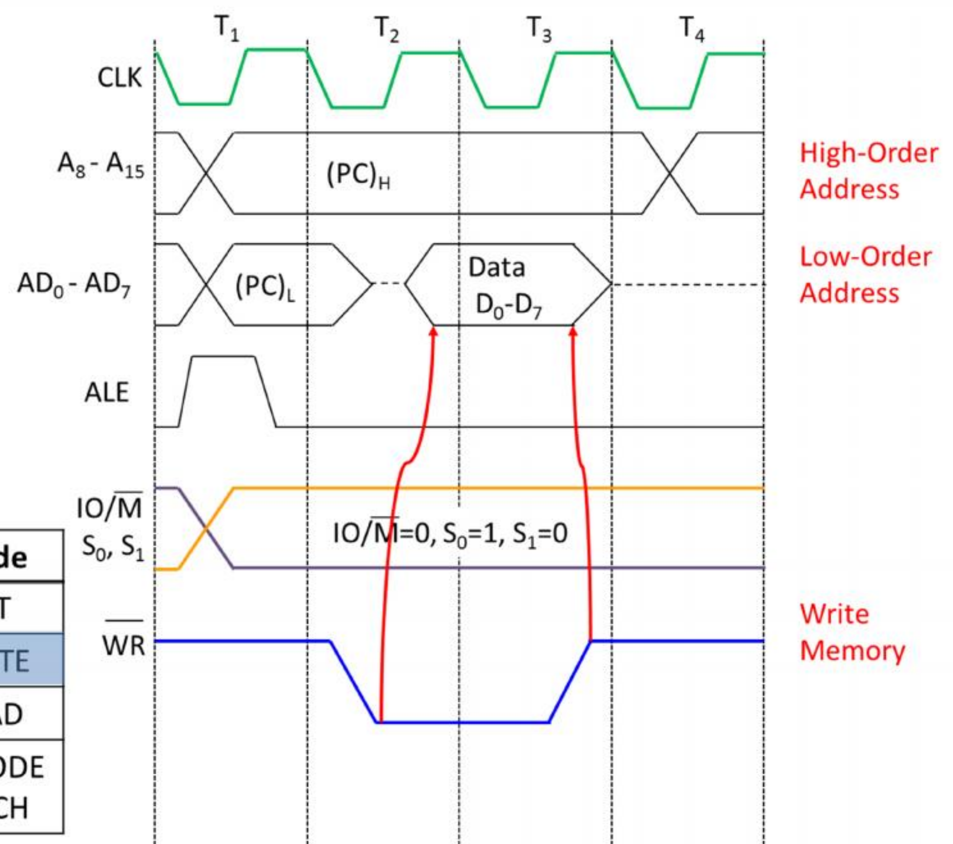


Figure: Memory Write Cycle.

- It is used to send one byte into memory.
- It requires 3 T-States.
- During T1, ALE is high and contains lower address A0-A7 from AD0-AD7.
- A8-A15 contains higher byte of address.
- As it is memory operation, IO/M (bar) goes low.
- During T2, ALE goes low, WR (bar) goes low and Address is removed from AD0-AD7 and then data appears on AD0-AD7.
- Data remains on AD0-AD7 till WR (bar) is low.

13. Explain how Control Signals Generated in 8085

Operation	IO/M'	RD'	WR'
MEMR'	0	0	X
MEMW'	0	X	0
IOR'	1	0	X
IOW'	1	X	0

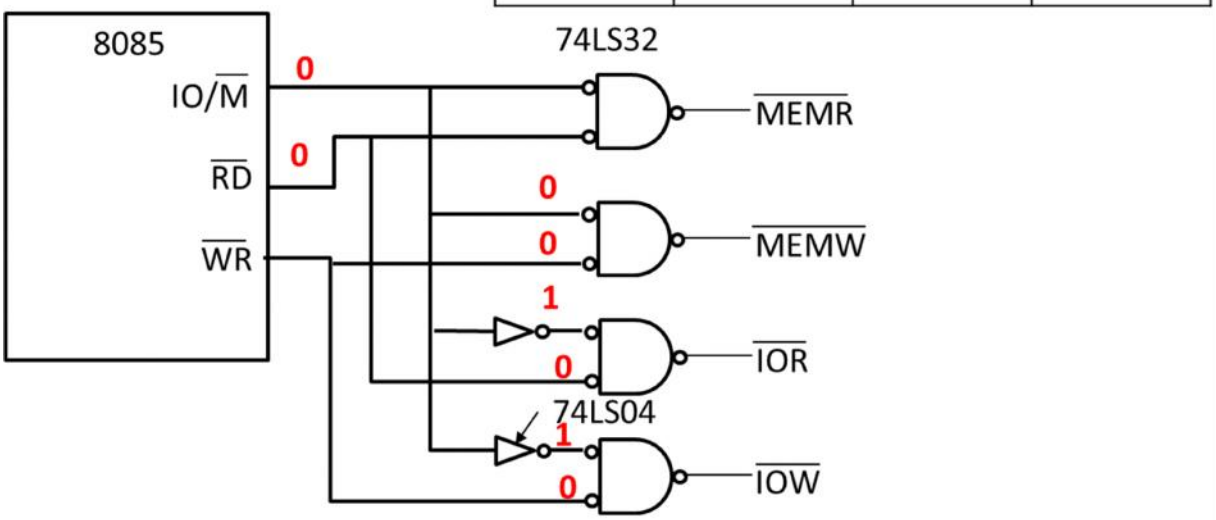


Figure: Control Signals Generated in 8085.

- Figure shows that four different control signals are generated by combining the signals RD (bar), WR (bar), and IO/M (bar).
- The signal IO/M (bar) goes low for the memory operation. This signal is ANDed with RD (bar) and WR (bar) signals b using the 74LS32 quadruple two-input OR gates, as shown in figure 4.5.
- The OR gates are functionally connected as negative NAND gates. When both input signals go low, the output of the gates go low and generate MEMR (bar) and MEMW (bar) control signals.
- When the IO/M (bar) signal goes high, it indicates the peripheral I/O operation.
- Figure shows that this signal is complemented using the Hex inverter 74LS04 and ANDed with the RD (bar) and WR (bar) signals to generate IOR (bar) and IOW (bar) control signals.

1. 8085 instruction set.

Sr.	Instruction	Description	Example
DATA TRANSFER INSTRUCTIONS			
1.	MOV R_d, R_s MOV M, R_s MOV R_s, M	This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers.	MOV B, C MOV B, M
2.	MVI Rd, data MVI M, data	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers.	MVI B, 57H MVI M, 57H
3.	LDA 16-bit address	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered.	LDA 2034H
4.	LDAX B/D Reg. pair	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered.	LDAX B
5.	LXI Reg.-pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand.	LXI H, 2034H LXI H, XYZ
6.	LHLD 16-bit address	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered.	LHLD 2040H
7.	STA 16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.	STA 4350H
8.	STAX Reg. pair	The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered.	STAX B

Sr.	Instruction	Description	Example
9.	SHLD 16-bit address	The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.	SHLD 2470H
10.	XCHG	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.	XCHG
11.	SPHL	The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.	SPHL
12.	XTHL	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered.	XTHL
13.	PUSH Reg. pair	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location.	PUSH B PUSH A
14.	POP Reg. pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1.	POP H POP A
15.	OUT 8-bit port address	The contents of the accumulator are copied into the I/O port specified by the operand.	OUT F8H
16.	IN 8-bit port address	The contents of the input port designated in the operand are read and loaded into the accumulator.	IN 8CH

ARITHMETIC INSTRUCTIONS

Sr.	Instruction	Description	Example
17.	ADD R ADD M	The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.	ADD B ADD M
18.	ADC R ADC M	The contents of the operand (register or memory) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.	ADC B ADC M
19.	ADI 8-bit data	The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.	ADI 45H
20.	ACI 8-bit data	The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.	ACI 45H
21.	DAD Reg. pair	The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected.	DAD H
22.	SUB R SUB M	The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.	SUB B SUB M
23.	SBB R SBB M	The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.	SBB B SBB M

Sr.	Instruction	Description	Example
24.	SUI 8-bit data	The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.	SUI 45H
25.	SBI 8-bit data	The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.	SBI 45H
26.	INR R INR M	The contents of the designated register or memory are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.	INR B INR M
27.	INX R	The contents of the designated register pair are incremented by 1 and the result is stored in the same place.	INX H
28.	DCR R DCR M	The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.	DCR B DCR M
29.	DCX R	The contents of the designated register pair are decremented by 1 and the result is stored in the same place.	DCX H
30.	DAA	The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.	DAA

Sr.	Instruction	Description	Example
BRANCHING INSTRUCTIONS			
31.	JMP 16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.	JMP 2034H JMP XYZ
Jump conditionally		*The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below.*	
32.	JC 16-bit address	Jump on Carry, Flag Status: CY=1	JC 2050H
33.	JNC 16-bit address	Jump on no Carry, Flag Status: CY=0	JNC 2050H
34.	JP 16-bit address	Jump on positive, Flag Status: S=0	JP 2050H
35.	JM 16-bit address	Jump on minus, Flag Status: S=1	JM 2050H
36.	JZ 16-bit address	Jump on zero, Flag Status: Z=1	JZ 2050H
37.	JNZ 16-bit address	Jump on no zero, Flag Status: Z=0	JNZ 2050H
38.	JPE 16-bit address	Jump on parity even, Flag Status: P=1	JPE 2050H
39.	JPO 16-bit address	Jump on parity odd, Flag Status: P=0	JPO 2050H
40.	CALL 16-bit address	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.	CALL 2034H CALL XYZ
Call conditionally		*The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.*	
41.	CC 16-bit address	Call on Carry, Flag Status: CY=1	CC 2050H
42.	CNC 16-bit address	Call on no Carry, Flag Status: CY=0	CNC 2050H
43.	CP 16-bit address	Call on positive, Flag Status: S=0	CP 2050H
44.	CM 16-bit address	Call on minus, Flag Status: S=1	CM 2050H
45.	CZ 16-bit address	Call on zero, Flag Status: Z=1	CZ 2050H
46.	CNZ 16-bit address	Call on no zero, Flag Status: Z=0	CNZ 2050H
47.	CPE 16-bit address	Call on parity even, Flag Status: P=1	CPE 2050H
48.	CPO 16-bit address	Call on parity odd, Flag Status: P=0	CPO 2050H

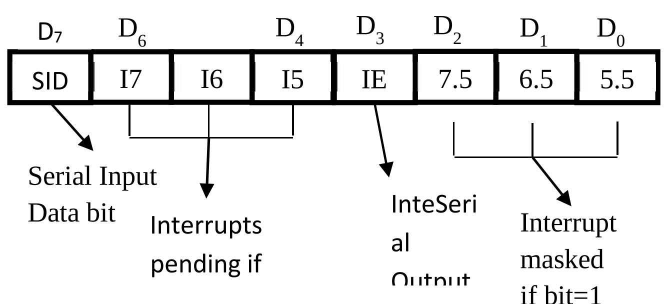
Sr.	Instruction	Description	Example
49.	RET	The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.	RET
	Return from subroutine conditionally	*The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.*	
50.	RC	Return on Carry, Flag Status: CY=1	RC
51.	RNC	Return on no Carry, Flag Status: CY=0	RNC
52.	RP	Return on positive, Flag Status: S=0	RP
53.	RM	Return on minus, Flag Status: S=1	RM
54.	RZ	Return on zero, Flag Status: Z=1	RZ
55.	RNZ	Return on no zero, Flag Status: Z=0	RNZ
56.	RPE	Return on parity even, Flag Status: P=1	RPE
57.	RPO	Return on parity odd, Flag Status: P=0	RPO
58.	PCHL	The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.	PCHL
59.	RST 0-7	The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addresses are:	RST 3

Instruction	**Restart Address**
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

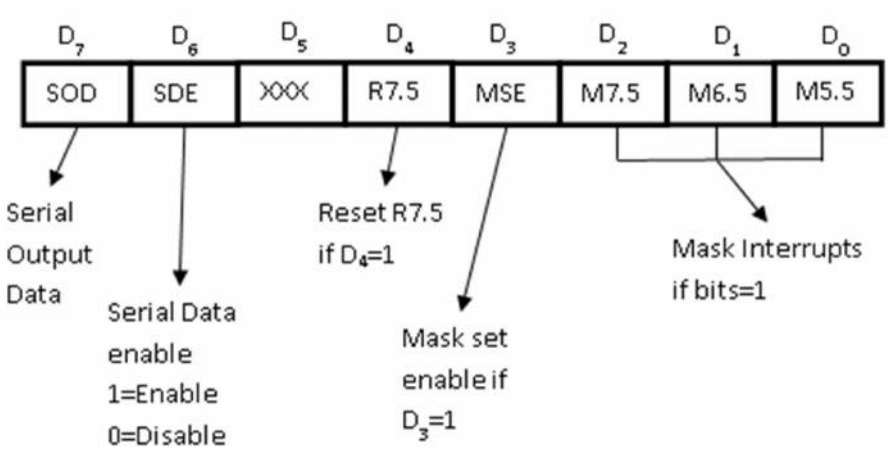
Sr.	Instruction	Description	Example
The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware.			
60.	TRAP	It restart from address 0024H	TRAP
61.	RST 5.5	It restart from address 002CH	RST 5.5
62.	RST 6.5	It restart from address 0034H	RST 6.5
63.	RST 7.5	It restart from address 003CH	RST 7.5
LOGICAL INSTRUCTIONS			
64.	CMP R CMP M	The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags of the PSW as follows: if (A) < (reg/mem): carry flag is set if (A) = (reg/mem): zero flag is set if (A) > (reg/mem): carry and zero flags are reset	CMP B CMP M
65.	CPI 8-bit data	The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows: if (A) < data: carry flag is set if (A) = data: zero flag is set if (A) > data: carry and zero flags are reset	CPI 89H
66.	ANA R ANA M	The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.	ANA B ANA M
67.	ANI 8-bit data	The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.	ANI 86H
68.	XRA R XRA M	The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	XRA B XRA M

Sr.	Instruction	Description	Example
69.	XRI 8-bit data	The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	XRI 86H
70.	ORA R ORA M	The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	ORA B ORA M
71.	ORI 8-bit data	The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.	ORI 86H
72.	RLC	Each binary bit of the accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. CY is modified according to bit D7. S, Z, P, AC are not affected.	RLC
73.	RRC	Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY is modified according to bit D0. S, Z, P, AC are not affected.	RRC
74.	RAL	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected.	RAL
75.	RAR	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected.	RAR
76.	CMA	The contents of the accumulator are complemented. No flags are affected.	CMA
77.	CMC	The Carry flag is complemented. No other flags are affected.	CMC
78.	STC	The Carry flag is set to 1. No other flags are affected.	STC
CONTROL INSTRUCTIONS			
79.	NOP	No operation is performed. The instruction is fetched and decoded. However no operation is executed.	NOP

Sr.	Instruction	Description	Example
80.	HLT	The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state.	HLT
81.	DI	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.	DI
82.	EI	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to re enable the interrupts (except TRAP).	EI
83.	RIM	This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.	RIM



Sr.	Instruction	Description	Example
84.	SIM	This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.	SIM



2. Explain Addressing mode in 8085

1) Immediate Addressing Mode

- In this mode 8/16 bit data is specified in instruction itself as one of its operand.
- Example

  MVI B 20H ; 20H is copied into register B.

  LXI D 1000H ; 1000H is stored into DE register pair.

2) Direct Addressing Mode

- In this mode 8/16 bit address is directly specified in instruction itself as one of its operand.
- Example

  LDA 2000H ; 2000H is memory address.

  IN 08H ; 08H is port address.

  OUT 10H ; 10H is port address.

3) Register Addressing Mode

- In this mode specifies register or register pair that contains data.
- Example

  MOV A B ; A ← B.

  ADD B ; A=A+B.

4) Indirect Addressing Mode

- In this mode 16 bit memory address is indirectly provided with the instruction using a register pair.
- Example

  LDAX D ; A ← M[DE].

  STAX D ; M[DE] ← A.

5) Implicit Addressing Mode

- This mode doesn't require any operand, data is specified by the Opcode itself.
- Example

  CMA

1. **Write an ALP to load register B with data 14H, register C with FFH, register D with 29H and register E with 67H.**

   MVI B, 14H

   MVI C, FFH

   MVI D, 29H

   MVI E, 67H

   HLT

2. **Write an ALP to transfer data from register B to C.**

   MVI B, 55H

   MOV C, B

   HLT

3. **Write an ALP to store data of register B into memory location 2050H.**

   MVI B, 67H

   MOV A, B

   STA 2050H ; Store data of Accumulator at memory location 2050H

   HLT

4. **write an ALP which directly store data 56H into memory location 2050H.**

   LXI H, 2050H

   MVI M, 56H

   HLT

5. **Write an 8085 assembly language program for exchanging two 8-bit numbers stored in memory locations 2050h and 2051h.**

   LDA 2050H

   MOV B, A

   LDA 2051H

   STA 2050H

   MOV A, B

   STA 2051H

   HLT

6. **Write an ALP to interchange 16-bit data stored in register BC and DE.**

   WITHOUT XCHG INSTRUCTION

   MOV H, B

MOV L, C

MOV B, D

MOV C, E

MOV D, H

MOV E, L

HLT

WITH XCHG INSTRUCTION

MOV H, B

MOV L, C

XCHG ; The contents of register H are exchanged with the contents of register D, and the

; contents of register L are exchanged with the contents of register E.

MOV B, H

MOV C, L

HLT

7. **Write the set of 8085 assembly language instructions to store the contents of B and C registers on the stack.**

MVI B, 50H

MVI C, 60H

PUSH B

PUSH C

HLT

8. **Write an ALP to delete (Make 00H) the data byte stored at memory location from address stores in register DE.**

MVI A, 00H

STAX D

HLT

9. **Write an 8085 assembly language program to add two 8-bit numbers stored in memory locations 2050h and 2051h. Store result in location 2052h.**

LXI H 2050H

MOV A M

INX H

ADD M

```
INX H
MOV M A
HLT
```

10. Subtract 8 bit data stored at memory location 2050H from data stored at memory location 2051H and store result at 2052H.

```
LXI H 2050H
MOV A M
INX H
SUB M ;  A = A - M
INX H
MOV M A
HLT
```

11. Write an 8085 assembly language program to add two 16-bit numbers stored in memory.

```
LHLD 2050H
XCHG        ; The contents of register H are exchanged with the contents of register D, and the
            ; contents of register L are exchanged with the contents of register E.
LHLD 2052H
MOV A E
ADD L
MOV L A
MOV A D
ADC H
MOV H A
SHLD 2054H ; Store Value of L Register at 2054 and value of H register at 2055.
HLT
```

12. Write an 8085 assembly language program to find the number of 1's binary representation of given 8-bit number.

```
MVI B 00H
MVI C 08H
MOV A D
```

```
BACK: RAR ; Rotate Accumulator Right through carry flag.
JNC SKIP
INR B
SKIP: DCR C ; Increment of B will skip.
JNZ BACK
HLT
```

13. Implement the Boolean equation D= (B+C) · E, where B, C, D and E represents data in various registers of 8085.

```
MOV A B
ORA C
ANA E
MOV D A
HLT
```

14. Write an 8085 assembly language program to add two decimal numbers using DAA instruction.

```
LXI H 2050H
MOV A M
INX H
MOV B M
MVI C 00H
ADD B
DAA ; Decimal adjustment of accumulator.
JNC SKIP
INR C
SKIP: INX H ; Increment of C will skip.
MOV M A
INX H
MOV M C
HLT
```

15. Write an 8085 assembly language program to find the minimum from two 8-bit numbers.

```
LDA 2050H
MOV B A
LDA 2051H
CMP B
JNC SMALL
STA 2052H
HLT
SMALL: MOV A B
STA 2052H
HLT
```

16. Write an 8085 program to copy block of five numbers starting from location 2001h to locations starting from 3001h.

```
LXI D 3100H
MVI C 05H
LXI H 2100
LOOP: MOV A M
STAX D
INX D
INX H
DCR C
JNZ LOOP
HLT
```

17. An array of ten data bytes is stored on memory locations 2100H onwards. Write an 8085 assembly language program to find the largest number and store it on memory location 2200H.

```
LXI H 2100H
MVI C 0AH
MOV A M
DCR C
LOOP: INX H
```

```
CMP M       ; Compare Data of accumulator with the data of memory location specified by HL pair and
            ; set flags accordingly.
JNC AHED
MOV A M
AHED: DCR C
JNZ LOOP
STA 2200H
HLT
```

18. Write an 8085 assembly language program to add block of 8-bit numbers.

```
LXI H 2000H
LXI B 3000H
LXI D 4000H
BACK: LDAX B
ADD M
STAX D
INX H
INX B
INX D
MOV A L
CPI 0A
JNZ BACK
HLT
```

19. Write an 8085 assembly language program to count the length of string ended with 0dh starting from location 2050h (Store length in register B).

```
LXI H 2050H
MVI B 00H
BACK: MOV A M
INR B
INX H
CPI 0DH
```

JNZ BACK

DCR B

HLT

20. An array of ten numbers is stored from memory location 2000H onwards. Write an 8085 assembly language program to separate out and store the EVEN and ODD numbers on new arrays from 2100H and 2200H, respectively.

```
LXI H 2000H
LXI D 2100H
LXI B 2200H
MVI A 0AH
COUNTER: STA 3000H
MOV A M
ANI 01H
JNZ CARRY
MOV A M
STAX B
INX B
JMP JUMP
CARRY: MOV A M ; This block will store Odd numbers.
STAX D
INX D
JUMP: LDA 3000H
DCR A
INX H
JNZ COUNTER
HLT
```

21. An array of ten data bytes is stored on memory locations 2100H onwards. Write an 8085 assembly language program to find the bytes having complemented nibbles (e.g. 2DH, 3CH, 78H etc.) and store them on a new array starting from memory locations 2200H onwards.

```
LXI H 2100H
LXI D 2200H
```

```
MVI C 0AH
LOOP: MOV A M
ANI 0FH
MOV B A
MOV A M
ANI F0H
RRC
RRC
RRC
RRC
CPM B
JNZ NEXT
MOV A M
STAX D
INX D
NEXT: INX H
DCR C
JNZ LOOP
HLT
```

22. Write an 8085 assembly language program to count the positive numbers, negative numbers, zeros, and to find the maximum number from an array of twenty bytes stored on memory locations 2000H onwards. Store these three counts and the maximum number on memory locations 3001H to 3004H, respectively.

```
LXI H 2000
MVI C 14
MVI D 00
MVI B 00
MVI E 00
LOOP: MOV A M
CMP B
```

```
JC NEG
JNZ POS
INX H
DCR C
JNZ LOOP
JMP STORE

NEG: INR D ; Count Negative number
INX H
DCR C
JNZ LOOP
JMP STORE

POS: INR E ; Count Positive number
INX H
DCR C
JNZ LOOP
JMP STORE

STORE: MOV A E
STA 3001

MOV A D
STA 3002

LXI H 2000
MVI C 14
MVI D 00
MVI B 00
```

```
MVI E 00

LOOP1: MOV A M ; Main Program for count Zero And Find Maximum.
CMP B
JZ ZERO
JNC MAX
INX H
DCR C
JNZ LOOP1
JMP STORE1

ZERO: INR D ; For count Zero
INX H
DCR C
JNZ LOOP1
JMP STORE1

MAX: CMP E ; Find Maximum.
JC SKIP
MOV E A
SKIP: INX H
DCR C
JNZ LOOP1
JMP STORE1

STORE1: MOV A D ; Store Number of zeros
STA 3003
MOV A E
STA 3004 ; Store maximum.
```

HLT

23. Write an 8085 assembly language program to separate out the numbers between 20_{10} and 40_{10} from an array of ten numbers stored on memory locations 2000H onwards. Store the separated numbers on a new array from 3000H onwards.

```
LXI H 2000

LXI D 3000

MVI C 0A

LOOP: MOV A M

CPI 14

JZ NEXT

JC NEXT

CPI 28

JNC NEXT

STAX D

INX D

NEXT: INX H ; Skip Storing of Number.

DCR C

JNZ LOOP

HLT
```

24. Write an 8085 assembly language program sort an array of twenty bytes stored on memory locations 2000H onwards in descending order.

```
MVI B 14

L2: LXI H 2000

MVI C 13

L1: MOV A M

INX H

CMP M

JC SWAP

bACK: DCR C

JNZ L1
```

DCR B

JNZ L2

HLT

SWAP: MOV D M; This block swap values.

MOV M A

DCX H

MOV M D

INX H

JMP BACK

25. An array of twenty data bytes is stored on memory locations 4100H onwards. Write an 8085 assembly language program to remove the duplicate entries from the array and store the compressed array on a new array starting from memory locations 4200H onwards.

MVI B 14H

MVI C 01H

LXI H 4101H

SHLD 3000H

LDA 4100H

STA 4200H

; This program fetch one by one value from original array and sore it on new array if it is not duplicate.

L1: LHLD 3000H

MOV A M

INX H

DCR B

JZ OVER

SHLD 3000H

LXI H 4200H

MOV D C

L2: CMP M

JZ L1

```
INX H
DCR D
JNZ L2
MOV M A
INR C
JMP L1
OVER: HLT
```

26. Write an ALP to Pack the two unpacked BCD numbers stored in memory locations 2200H and 2201H and store result in memory location 2300H. Assume the least significant digit is stored at 2200H.

```
LDA 2201
RLC ; Rotate accumulator left 4 times without carry.
RLC
RLC
RLC
ANI F0
MOV C A
LDA 2200
ADD C
STA 2300
HLT
```

27. Write a set of 8085 assembly language instructions to unpack the upper nibble of a BCD number.

```
MVI A 98
MOV B A
ANI F0
RRC ; Rotate accumulator left 4 times without carry.
RRC
RRC
RRC
STA 2000
```

HLT

28. Write Assembly language program to subtract 2 16-bit BCD numbers.

```
LXI H 3040
LXI D 1020
MOV A L
SUB E
DAA
STA 2000
MOV A H
SBB D
DAA
STA 2001
HLT
```

29. Write an 8085 assembly language program to continuously read an input port with address 50H. Also write an ISR to send the same data to output port with address A0H when 8085 receives an interrupt request on its RST 5.5 pin.

```
LOOP: IN 50
EI
CALL DELAY
JMP LOOP
HLT
DELAY: NOP
NOP
NOP
NOP
RET


; This code must be write at memory location 002C onwards.
OUT A0
JMP LOOP
```

30. Write an ALP to generate a square wave of 2.5 kHz frequency. Use D_0 bit of output port ACH to output the square wave.

MVI A 01H

REPEAT: OUT AC

MVI C Count

AGAIN: DCR C

JNZ AGAIN

CMA

JMP REPEAT

Calculation:

$$Time\ period\ of\ square\ wave = \frac{1}{2.5 * 10^3} = 0.4 * 10^{-3}s.$$

$$Time\ period\ of\ upper\ half\ and\ lower\ half\ of\ square\ wave = \frac{0.4 * 10^{-3}s}{2}. = 0.2 * 10^{-3}s.$$

$$let\ processor\ time\ period = 0.3 * 10^{-6}s.$$

$$Delay\ required\ beween\ transition\ of\ square\ wave = \frac{0.2 *\ 10^{-3}}{0.3 * 10^{-6}} \approx 666 Tstates$$

Now

$666 = 7 + (14 * Count) - 3 + 4$

$658 = 14 * Count$

$Count = 47$

$Count = 2FH$

Final Program:

MVI A 01H

REPEAT: OUT AC

MVI C 2F

AGAIN: DCR C

JNZ AGAIN

CMA

JMP REPEAT

1. Stack

- Stack is a group of memory location in the R/W memory that is used for temporary storage of binary information during execution of a program.
- The starting memory location of the stack is defined in program and space is reserved usually at the high end of memory map.
- The beginning of the stack is defined in the program by using instruction **LXI SP, 16-bit memory address**. Which loads a 16-bit memory address in stack pointer register of microprocessor.
- Once stack location is defined storing of data bytes begins at the memory address that is one less then address in stack pointer register. LXI SP, 2099h the storing of data bytes begins at 2098H and continues in reversed numerical order.

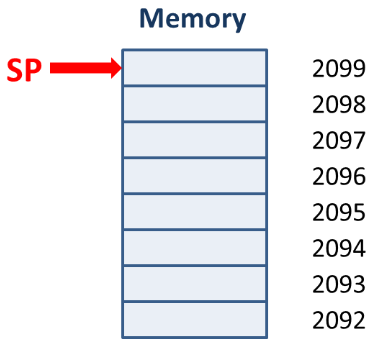


Fig. Stack

- Data bytes in register pair of microprocessor can be stored on the stack in reverse order by using the PUSH instruction.
- PUSH B instruction sore data of register pair BC on sack.

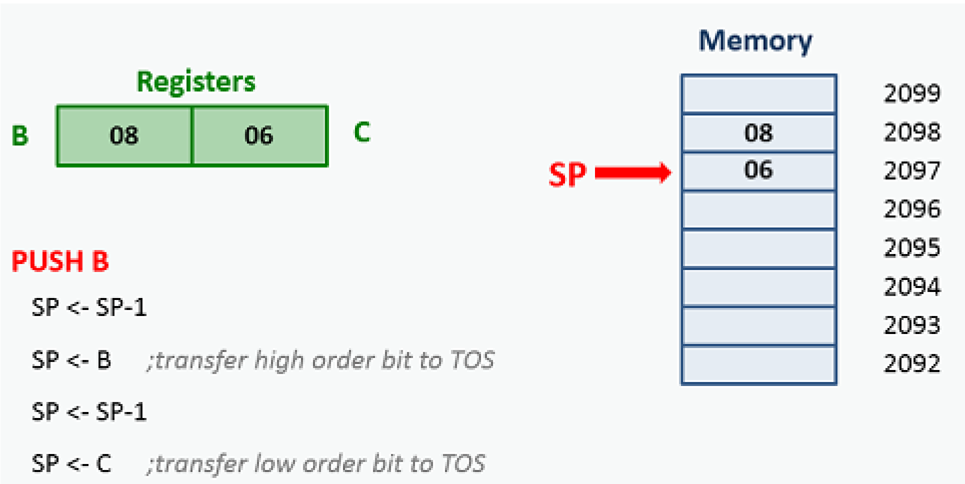


Fig. PUSH operation on stack

- Data bytes can be transferred from the stack to respective registers by using instruction POP.

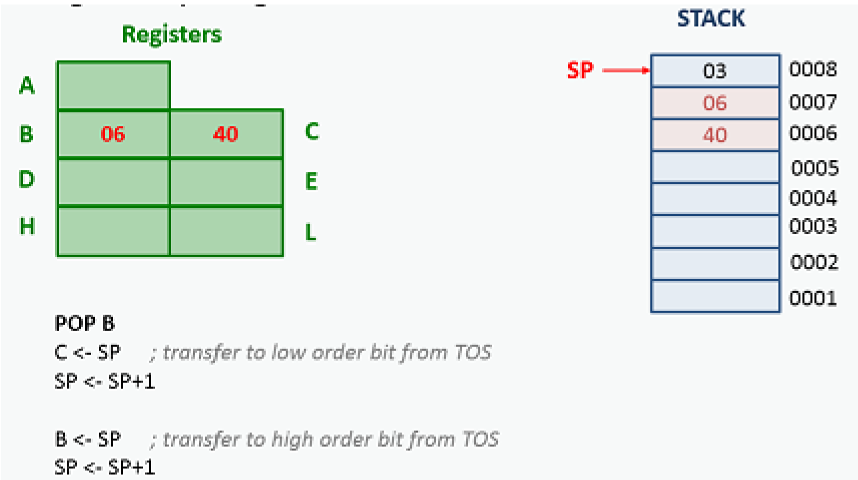


Fig. POP operation on stack

Instruction necessary for stack in 8085

LXI SP, 2095	Load the stack pointer register with a 16-bit address.
PUSH B/D/H	It copies contents of B-C/D-E/H-L register pair on the stack.
PUSH PSW	Operand PSW represents Program status word meaning contents of accumulator and flags.
POP B/D/H	It copies content of top two memory locations of the stack in to specified register pair.
POP PSW	It copies content of top two memory locations of the stack in to B-C accumulator and flags respectively.

2. Subroutine

- A subroutine is a group of instruction that performs a subtask of repeated occurrence.
- A subroutine can be used repeatedly in different locations of the program.

Advantage of using Subroutine

- Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations.

Where to write Subroutine?

- In Assembly language, a subroutine can exist anywhere in the code.
- However, it is customary to place subroutines separately from the main program.

Instructions for dealing with subroutines in 8085.

- The **CALL** instruction is used to redirect program execution to the subroutine.
  - When CALL instruction is fetched, the Microprocessor knows that the next two **new** Memory location contains 16bit subroutine address.
  - Microprocessor Reads the subroutine address from the next two memory location and stores the higher order 8bit of the address in the **W** register and stores the lower order 8bit of the address in the **Z** register.
  - Push the **Older** address of the instruction immediately following the CALL onto the stack [Return address]
  - Loads the program counter (**PC**) with the **new** 16-bit address supplied with the CALL instruction from **WZ** register.
- The **RET** instruction is used to return.

- Number of PUSH and POP instruction used in the subroutine must be same, otherwise, RET instruction will pick wrong value of the return address from the stack and program will fail.

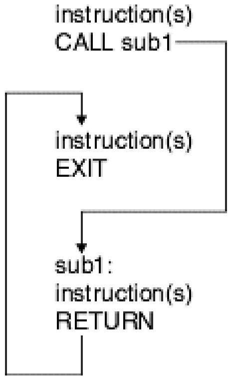


Fig. Subroutine

- Example: write ALP to add two numbers using call and subroutine.

```
LXI H 2000 ; Load memory address of operand
MOV B M ; Store first operand in register B
INX H ;Increment H-L pair
MOV A M ; Store second operand in register A
CALL ADDITION ; Call subroutine ADDITION
STA 3000 ; Store answer
HLT

ADDITION: ADD B ; Add A and B
RET ; Return
```

Conditional call and return instruction available in 8085

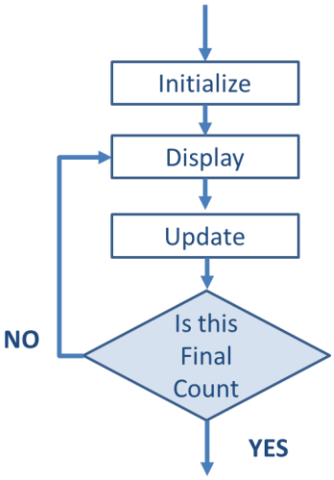
CC 16-bit address	Call on Carry, Flag Status: CY=1
CNC 16-bit address	Call on no Carry, Flag Status: CY=0
CP 16-bit address	Call on positive, Flag Status: S=0
CM 16-bit address	Call on minus, Flag Status: S=1
CZ 16-bit address	Call on zero, Flag Status: Z=1
CNZ 16-bit address	Call on no zero, Flag Status: Z=0
CPE 16-bit address	Call on parity even, Flag Status: P=1
CPO 16-bit address	Call on parity odd, Flag Status: P=0
RC	Return on Carry, Flag Status: CY=1
RNC	Return on no Carry, Flag Status: CY=0
RP	Return on positive, Flag Status: S=0
RM	Return on minus, Flag Status: S=1
RZ	Return on zero, Flag Status: Z=1
RNZ	Return on no zero, Flag Status: Z=0
RPE	Return on parity even, Flag Status: P=1
RPO	Return on parity odd, Flag Status: P=0

3. Applications of Counters and Time Delays

1. Traffic Signal
2. Digital Clocks
3. Process Control
4. Serial data transfer

4. Counters

- A counter is designed simply by loading appropriate number into one of the registers and using INR or DNR instructions.
- Loop is established to update the count.
- Each count is checked to determine whether it has reached final number; if not, the loop is repeated.




```
1. MVI C,05
2. LOOP: MOV A,C
3.       OUT 01
4.       DCR C
5.    JNZ LOOP
6. HLT
```

Fig. Counter

5. Time Delay

- Each instruction passes through different combinations of Fetch, Memory Read, and Memory Write cycles.
- Knowing the combinations of cycles, one can calculate how long such an instruction would require to complete.
- It is counted in terms of number of T–states required.
- Calculating this time we generate require software delay.

Time Delay Using Single Register

Label	Opcode	Operand	Comment	T-states
	MVI	C,05h	; Load Counter	**7**
LOOP:	DCR	C	; Decrement Counter	**4**
	JNZ	LOOP	; Jump back to Decr. C	**10/7**

MVI C 05 Mchine Cycle: F + R = 2 T-States: 4T + 3T = 7T	DCR C Mchine Cycle: F = 1 T-States: 4T = 4T	JNZ LOOP (true) Mchine Cycle: F + R + R = 3 T-States: 4T + 3T + 3T = 10T	JNZ LOOP (false) Mchine Cycle: F + R = 3 T-States: 4T + 3T = 7T

- Instruction MVI C, 05h requires 7 T-States to execute. Assuming, 8085 Microprocessor with 2MHz clock frequency. How much time it will take to execute above instruction?
  Clock frequency of the system (f) = 2 MHz
  Clock period (T) = 1/f = ½ * 10-6 = 0.5 µs
  Time to execute MVI = 7 T-states * 0.5 µs
  = 3.5 µs
- Now to calculate time delay in loop, we must account for the T-states required for each instruction, and for the number of times instructions are executed in the loop. There for the next two instructions:
  DCR: 4 T-States
  JNZ: 10 T-States
  14 T-States
- Here, the loop is repeated for 5 times.
- Time delay in loop T_L with 2MHz clock frequency is calculated as:
  T_L= T * Loop T-sates * N_{10} -----------------(1)
  T_L : Time Delay in Loop
  T : Clock Frequency
  N_{10} : Equivalent decimal number of hexadecimal count loaded in the delay register.
- Substituting value in equation (1)
  T_L= (0.5 * 10^{-6} * 14 * 5)
  = 35 µs
- If we want to calculate delay more accurately, we need to accurately calculate execution of JNZ instruction i.e
  If **JNZ = true**, then **T-States = 10**
  Else if **JNZ =false**, then **T-States = 7**
- Delay generated by last clock cycle:
  = 3T * Clock Period
  = 3T * (1/2 * 10^{-6})
  = 1.5 µs
- Now, the accurate loop delay is:
  T_{LA}=T_L - Delay generated by last clock cycle
  T_{LA}= 35 µs - 1.5 µs
  T_{LA}= 33.5 µs
- Now, to calculate total time delay
  Total Delay = Time taken to execute instruction outside loop + Time taken to execute loop instructions
  T_D = T_O + T_{LA}
  = (7 * 0.5 µs) + 33.5 µs
  = 3.5 µs + 33.5 µs
  = 37 µs
- In most of the case we are given time delay and need to find value of the counter register which decide number of times loop execute.
- For example: write ALP to generate 37 µs delay given that clock frequency if 2 MHz.
- Single register loop can generate small delay only for large delay we use other technique.

Time Delay Using a Register Pair

- Time delay can be considerably increased by setting a loop and using a register pair with a 16-bit number (FFFF h).
- A 16-bit is decremented by using DCX instruction.
- Problem with DCX instruction is DCX instruction doesn't set **Zero** flag.
- Without test flag, Jump instruction can't check desired conditions.
- Additional technique must be used to set Zero flag.

Label	Opcode	Operand	Comment	T-states
	LXI	B,2384 h	; Load BC with 16-bit counter	**10**
LOOP:	DCX	B	; Decrement BC by 1	**6**
	MOV	A, C	; Place contents of C in A	**4**
	ORA	B	; OR B with C to set Zero flag	**4**
	JNZ	LOOP	; if result not equal to 0, 10/7 jump back to loop	**10/7**

- Here the loop includes four instruction:
  Total T-States = 6T + 4T + 4T + 10T
  = 24 T-states
- The loop is repeated for 2384 h times.
- Converting $(2384)_{16}$ into decimal.
  2384 h = $(2 * 16^3) + (3 * 16^2) + (8 * 16^1) + (4 * 16^0)$
  = 8192 + 768 + 128 + 4 = **9092**
- Clock frequency of the system (f)= 2 MHz
- Clock period (T) = 1/f = ½ * 10^{-6} = 0.5 μs
- Now, to find delay in the loop
  T_L= T * Loop T-sates * N_{10}
  = 0.5 * 24 * 9092
  = 109104 μs = 109 ms (without adjusting last cycle)

Time Delay Using a LOOP within a LOOP

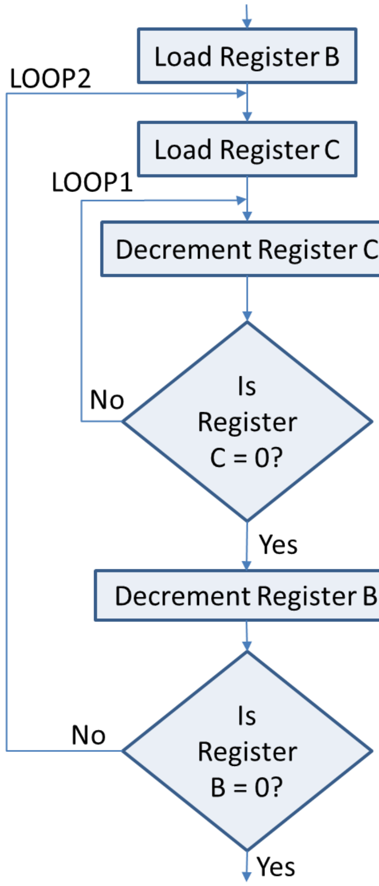


Fig. Time Delay Using a LOOP within a LOOP

Label	Opcode	Operand	T-states
	MVI	B,38h	7T
LOOP2:	MVI	C,FFh	7T
LOOP1:	DCR	C	4T
	JNZ	LOOP1	10/7 T
	DCR	B	4T
	JNZ	LOOP2	10/7 T

- Calculating delay of inner LOOP1: T_{L1}

  T_L= T * Loop T-states * N_{10}

  = 0.5 * 14* 255

  = 1785 µs = 1.8 ms

  T_{L1}= TL – (3T states* clock period)

  = 1785 – (3 * ½ * 10^{-6})

  = 1785-1.5=**1783.5 µs**
- Now, Calculating delay of outer LOOP2: T_{L2}
- Counter B : $(38)_{16}$ = **$(56)_{10}$** So loop2 is executed for 56 times.

  T-States = 7 + 4 + 10 = 21 T-States

  T_{L2} = 56 (T_{L1} + 21 T-States * 0.5)

  = 56(1783.5 µs + 10.5)

= 100464 µs

T_{L2} = 100.46 ms

Disadvantage of using software delay

- Accuracy of time delay depends on the accuracy of system clock.
- The Microprocessor is occupied simply in a waiting loop; otherwise it could be employed to perform other functions.
- The task of calculating accurate time delays is tedious.
- In real time applications timers (integrated timer circuit) are commonly used.
- Intel 8254 is a programmable timer chip that can be interfaced with microprocessor to provide timing accuracy.
- The disadvantage of using hardware chip include the additional expense and the need for extra chip in the system.

6. Counter design with time delay

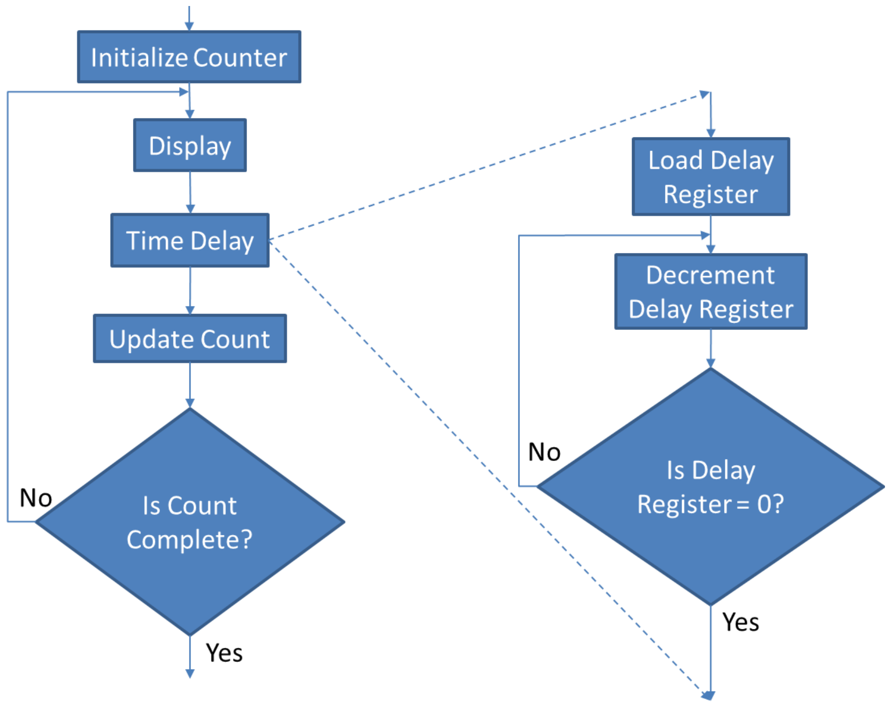


Fig. 6. Counter design with time delay

- It is combination of counter and time delay.
- I consist delay loop within counter program.

7. Hexadecimal counter program

- Write a program to count continuously in hexadecimal from **FFh** to **00h** with **0.5 µs** clock period. Use register C to set up 1 ms delay between each count and display the number at one of the output port.
- **Given:**
- Counter= FF h
- Clock Period T=0.5 µs

- Total Delay = 1ms
- **Output:**
- To find value of delay counter
- **Program**

```
MVI B,FF
LOOP:MOV A,B
    OUT 01
    MVI C, COUNT; need to calculate delay count
    DELAY: DCR C
        JNZ DELAY
    DCR B
    JNZ LOOP
HLT
```

- Calculate Delay for Internal Loop
  TI = T-States * Clock Period * COUNT
  = 14 * 0.5 * 10-6 * COUNT
  TI = (7.0 * 10-6)* COUNT
- Calculate Delay for Outer Loop:
  TO = T-States * Clock Period
  = 35 * 0.5 * 10-6
  TO = 17.5 ?s
- Calculate Total Time Delay:
  TD = TO + TL
  1 ms = 17.5 * 10-6 + (7.0 * 10-6)* COUNT
  1 * 10-3 = 17.5 * 10-6 + (7.0 * 10-6)* COUNT
  COUNT="1 ∗ 10−3 − 17.5 ∗ 10−6" /"7.0 ∗ 10−6"
  COUNT= $(140)_{10}$ = $(8C)_{16}$

8. 0-9 up/down counter program

- Write an 8085 assembly language program to generate a decimal counter (which counts 0 to 9 continuously) with a one second delay in between. The counter should reset itself to zero and repeat continuously. Assume a crystal frequency of 1MHz.
- **Program**

```
START: MVI B,00H
DISPLAY: OUT 01
    LXI H, COUNT
LOOP: DCX H
    MOV A, L
    ORA H
    JNZ LOOP
    INR B
    MOV A,B
    CPI 0A
    JNZ DISPLAY
```

JZ START

9. Code Conversion

Two Digit BCD Number to Binary Number

1. Initialize memory pointer to given address (2000).
2. Get the Most Significant Digit (MSD).
3. Multiply the MSD by ten using repeated addition.
4. Add the Least Significant Digit (LSD) to the result obtained in previous step.
5. Store the HEX data in Memory.

- **Program**

```
LXI H 2000
MOV C M
MOV A C
ANI 0F ; AND operation with 0F (00001111)
MOV E A
MOV A C
ANI F0 ; AND operation with F0 (11110000)
JZ SB1 ; If zero skip further process and directly add LSD
RRC ; Rotate 4 times right
RRC
RRC
RRC
MOV D A
MVI A 00
L1: ADI 0A ; Loop L1 multiply MSD with 10
DCR D
JNZ L1
SB1: ADD E
STA 3000 ; Store result
HLT
```

8-bit Binary Number to Decimal Number

1. Load the binary data in accumulator
2. Compare ‘A’ with 64 (Dicimal 100) if cy = 01, go step 5 otherwise next step
3. Subtract 64H from ‘A’ register
4. Increment counter 1 register
5. Go to step 2
6. Compare the register ‘A’ with ‘0A’ (Dicimal 10), if cy=1, go to step 10, otherwise next step
7. Subtract 0AH from ‘A’ register
8. Increment Counter 2 register
9. Go to step 6
10. Combine the units and tens to from 8 bit result
11. Save the units, tens and hundred’s in memory
12. Stop the program execution

- **Program**

```
MVI B 00
```

```
LDA 2000
LOOP1: CPI 64 ; Compare with 64H
JC NEXT1 : If A is less than 64H then jump on NEXT1
SUI 64 ; subtract 64H
INR B
JMP LOOP1
NEXT1:  LXI H 2001
MOV M B ; Store MSD into memory
MVI B 00
LOOP2: CPI 0A ; Compare with 0AH
JC NEXT2 ; If A is less than 0AH then jump on NEXT2
SUI 0A ; subtract 0AH
INR B
JMP LOOP2
NEXT2:  MOV D A
MOV A B
RLC
RLC
RLC
RLC
ADD D
STA 2002 ; Store packed number formed with two leas significant digit
HLT
```

Binary Number to ASCII Number

- Load the given data in A - register and move to B - register
- Mask the upper nibble of the Binary decimal number in A - register
- Call subroutine to get ASCII of lower nibble
- Store it in memory
- Move B - register to A - register and mask the lower nibble
- Rotate the upper nibble to lower nibble position
- Call subroutine to get ASCII of upper nibble
- Store it in memory
- Terminate the program.

```
LDA 5000 Get Binary Data
    MOV B, A
    ANI 0F        ; Mask Upper Nibble
    CALL SUB1     ; Get ASCII code for upper nibble
    STA 5001
    MOV A, B
    ANI F0        ; Mask Lower Nibble
    RLC
    RLC
    RLC
    RLC
    CALL SUB1     ; Get ASCII code for lower nibble
    STA 5002
```

```
      HLT        ; Halt the program.

SUB1:  CPI 0A
      JC SKIP
      ADI 07
SKIP:  ADI 30
      RET        ; Return Subroutine
```

ASCII Character to Hexadecimal Number

1. Load the given data in A - register
2. Subtract 30H from A - register
3. Compare the content of A - register with 0AH
4. If A < 0AH, jump to step6. Else proceed to next step
5. Subtract 07H from A - register
6. Store the result
7. Terminate the program

- **Program**

```
LDA 2000
CALL ASCTOHEX
STA 2001
HLT

ASCTOHEX: SUI 30 ; This block Convert ASCII to Hexadecimal.
CPI 0A
RC
SUI 07
RET
```

10. BCD Arithmetic

Add 2 8-bit BCD Numbers

1. Load firs number into accumulator.
2. Add second number.
3. Apply decimal adjustment to accumulator.
4. Store result.

- **Program**

```
LXI H, 2000H
MOV A, M
INX H
ADD M
DAA
INX H
MOV M, A
HLT
```

Subtract the BCD number stored in E register from the number stored in the D register

1. Find 99's complement of data of register E
2. Add 1 to find 100's complement of data of register E
3. Add Data of Register D
4. Apply decimal adjustment

- **Program**

```
MVI A, 99H
SUB E        : Find the 99's complement of subtrahend
INR A        : Find 100's complement of subtrahend
ADD D        : Add minuend to 100's complement of subtrahend
DAA          : Adjust for BCD
HLT          : Terminate program execution
```

11. 16-Bit Data operations

Add Two 16 Bit Numbers

1. Initialize register C for using it as a counter for storing carry value.
2. Load data into HL register pair from one memory address (9000H).
3. Exchange contents of register pair HL with DE.
4. Load second data into HL register pair (from 9002H).
5. Add register pair DE with HL and store the result in HL.
6. If carry is present, go to 7 else go to 8.
7. Increment register C by 1.
8. Store value present in register pair HL to 9004H.
9. Move content of register C to accumulator A.
10. Store value present in accumulator (carry) into memory (9006H).
11. Terminate the program.

- **Program**

```
MVI C, 00H
LHLD 9000H
XCHG ; Exchange contents of register pair HL with DE
LHLD 9002H
DAD D ; Add register pair DE with HL and store the result in HL
JNC AHEAD ; If carry is present, go to AHEAD
INR C
AHEAD: SHLD 9004H ; Store value present in register pair HL to 9004H
MOV A, C
STA 9006H ; Store value present in accumulator (carry) into memory (9006H)
HLT
```

Subtract Two 16 Bit Numbers

1. Load first data from Memory (9000H) directly into register pair HL.
2. Exchange contents of register pair DE and HL.
3. Load second data from memory location (9002H) directly into register pair HL.

4. Move contents of register E into accumulator A.
5. Subtract content of register L from A.
6. Move contents of accumulator A into register L.
7. Move contents of register D into accumulator A.
8. Subtract with borrow contents of register H from accumulator A.
9. Move contents of accumulator A into register H.
10. Store data contained in HL register pair into memory (9004H).
11. Terminate the program.

- **Program**

```
LHLD 9000H ; Load first data from Memory (9000H) directly into register pair HL
XCHG ; Exchange contents of register pair DE and HL.
LHLD 9002H ; Load second data from memory location (9002H) directly into register pair HL
MOV A, E
SUB L
MOV L, A
MOV A, D
SBB H ; Subtract with borrow contents of register H from accumulator A
MOV H, A
SHLD 9004H ; Store data contained in HL register pair into memory (9004H)
HLT
```

1. Draw and explain the block diagram of the programmable peripheral interface (8255A).

Ans.

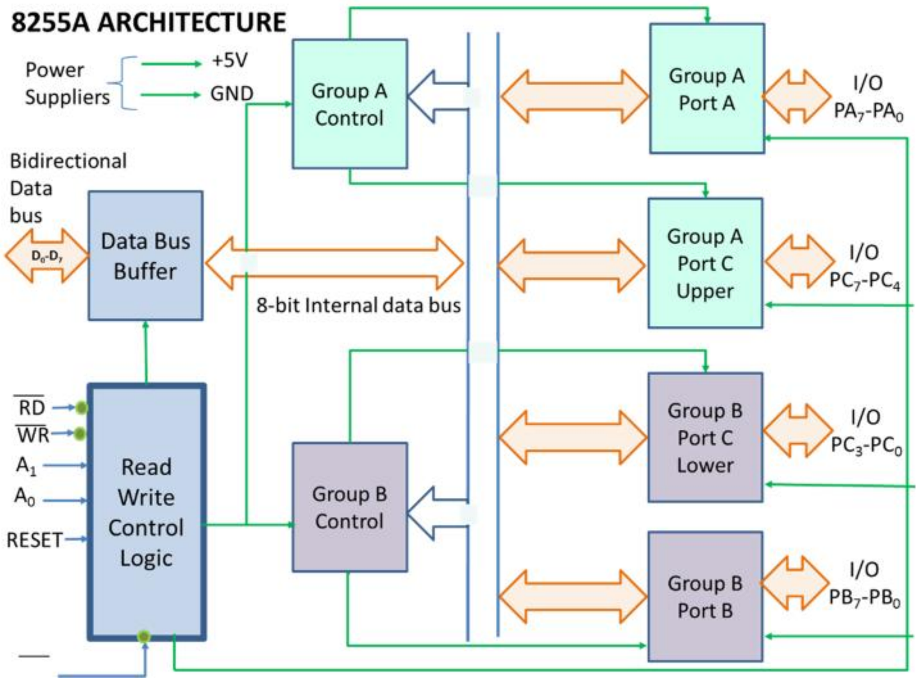


Figure: 8255A Architecture

Read Write Control Logic

RD (READ)	This is an active low signal that enables Read operation. When signal is low MPU reads data from selected I/O port of 8255A
WR (WRITE)	This is an active low signal that enables Write operation. When signal is low MPU writes data into selected I/O port or control register
RESET	This is an active high signal, used to reset the device. That means clear control registers
CS	This is Active Low signal. When it is low, then data is transfer from 8085 CS signal is the master Chip Select. A_0 and A_1 specify one of the I/O ports or control register

CS	A1	A0	Selected
0	0	0	PORT A
0	0	1	PORT B
0	1	0	PORT C
0	1	1	Control Register
1	X	X	8255A is not selected

Data Bus Buffer

- This three-state bi-directional 8-bit buffer is used to interface the 8255 to the system data bus.
- Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU.
- Control words and status information are also transferred through the data bus buffer.

Group A and Group B Controls

- The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255.
- The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255.
- Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Ports A, B, and C

- The 8255 contains three 8-bit ports (A, B, and C).
- All can be configured to a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255.
- Port A One 8-bit data output latch/buffer and one 8-bit data input latch.
- Both "pull-up" and "pull-down" bus-hold devices are present on Port A.
- Port B One 8-bit data input/output latch/buffer and one 8-bit data input buffer.
- Port C One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control.
- Each 4-bit port contains a 4-bit latch and it can be used for the control signal output and status signal inputs in conjunction with ports A and B.

2. Explain 8255A I/O Operating Modes

Ans. 8255A has three different I/O operating modes:

1. Mode 0
2. Mode 1
3. Mode 2

Mode 0

- Simple I/O for port A,B and C
- In this mode, Port A and B is used as two 8-bit ports and Port C as two 4-bit ports.
- Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched.
- Ports do not have handshake or interrupt capability.

Mode 1: Input or Output with Handshake

- Handshake signal are exchanged between MPU and peripheral prior to data transfer.
- In this mode, Port A and B is used as 8-bit I/O ports.
- Mode 1 is a handshake Mode whereby ports A and/or B use bits from port C as handshake signals.
- In the handshake mode, two types of I/O data transfer can be implemented: status check and interrupt.
- Port A uses upper 3 signals of Port C: PC3, PC4, PC5
- Port B uses lower 3 signals of Port C : PC0, PC1, PC2
- PC6 and PC7 are general purpose I/O pins

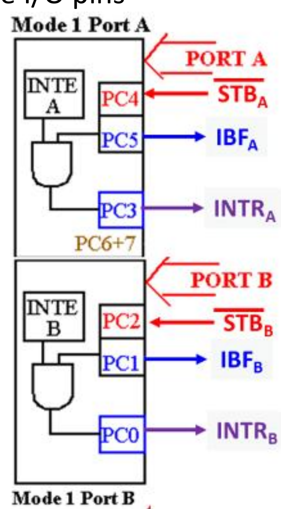


Figure: Mode1 Input Handshake

STB (Strobe Input):

- This active low signal is generated by a peripheral device to indicate that, it has transmitted a byte of data. The 8255A, in response to **STB**, generates **IBF** and **INTR**.

IBF (Input Buffer Full)

This signal is acknowledged by 8255A to indicate that the input latch has received the data byte. It will get reset when the MPU reads the data.

INTR(Interrupt Request)

This is an output signal that may be used to interrupt the MPU. This signal is generated if STB, IBF and INTE (internal flip-flop) are all at logic 1. It will get reset by the falling edge of RD

INTE(Interrupt Enable)

- This signal is an internal flip-flop, used to enable or disable the generation of INTR signal.
- The interrupt enable signal is neither an input nor an output; it is an internal bit programmed via the PC4 (port A) or PC2 (port B) bits.

Mode 2

- In this mode, Port A can be configured as the bidirectional port and Port B either in Mode 0 or Mode 1.
- Port A uses five signals from Port C as handshake signals for data transfer.
- The remaining three signals from Port C can be used either as simple I/O or as handshake for port B.

3. Explain BSR Mode of the programmable peripheral interface (8255A) with necessary diagrams.

Ans.

- These are two basic modes of operation of 8255.
  I/O mode and Bit Set-Reset mode (BSR).
- In I/O mode, the 8255 ports work as programmable I/O ports, while in BSR mode only port C (PC0-PC7) can be used to set or reset its individual port bits.
- Under the I/O mode of operation, further there are three modes of operation of 8255, so as to support different types of applications, mode 0, mode 1 and mode 2.

8255A: BSR(Bit Set/Reset) Mode

- In this mode any of the 8-bits of port C can be set or reset depending on D_0 of the control word.
- The bit to be set or reset is selected by bit select flags D_3, D_2 and D_1 of the CWR (Control Word Register).
- BSR Control Word affects one bit at a time
- It does not affect the I/O mode

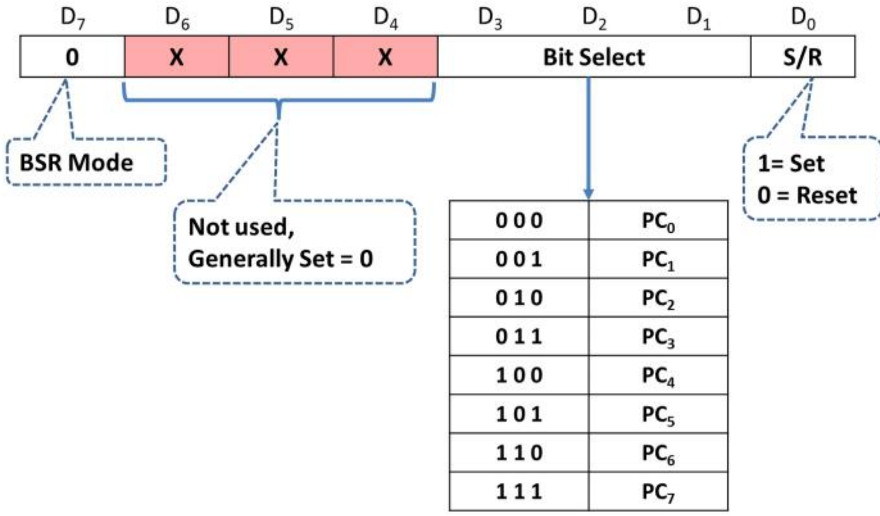


Figure: BSR Mode Control Word

4. Explain 8255A Control Word and Control Register with necessary diagram.

Ans. Control Register

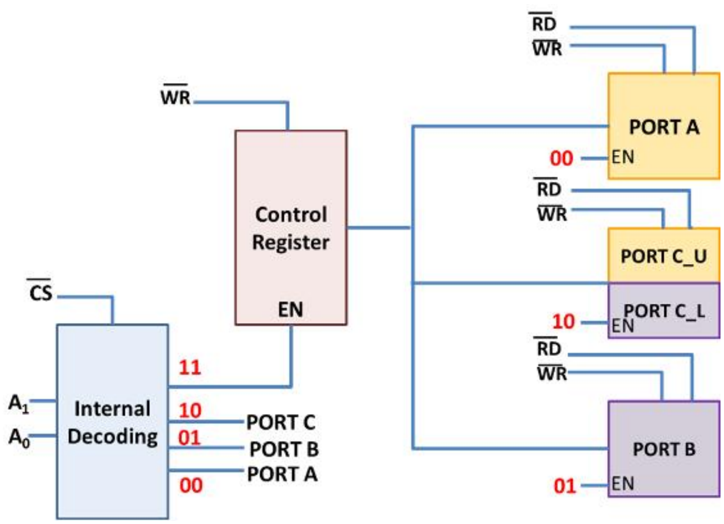


Figure: Control Register 8255A

Control Word: Content of Control register is known as Control Word.

- Control word specify an I/O function for each port this register can be.

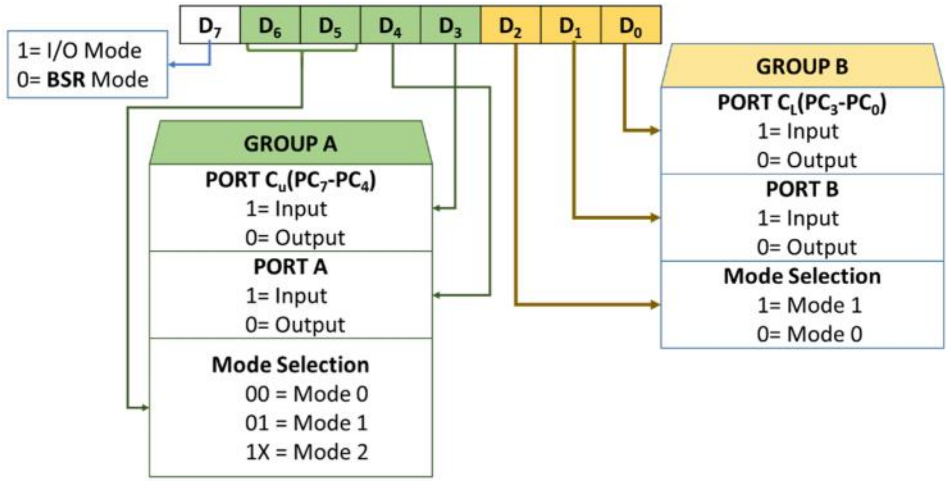


Figure:8255A Control Word

- Accessed to write a control word when A0 and A1 are at logic1, the register is not accessible for a read operation.
- Bit D7 of the control register either specifies the I/O function or the bit Set/Reset function, as classified in figure 1.
- If bit D7=0, bits D6-D0 determine I/O function in various mode, as shown in figure 4.
- If bit D7=0 port C operates in the bit Set/Reset (BSR) mode.
- The BSR control word does not affect the function of port A and B.

5. What is the need of the programmable interrupt controller (8259A)? Draw and explain the block diagram of 8259A.

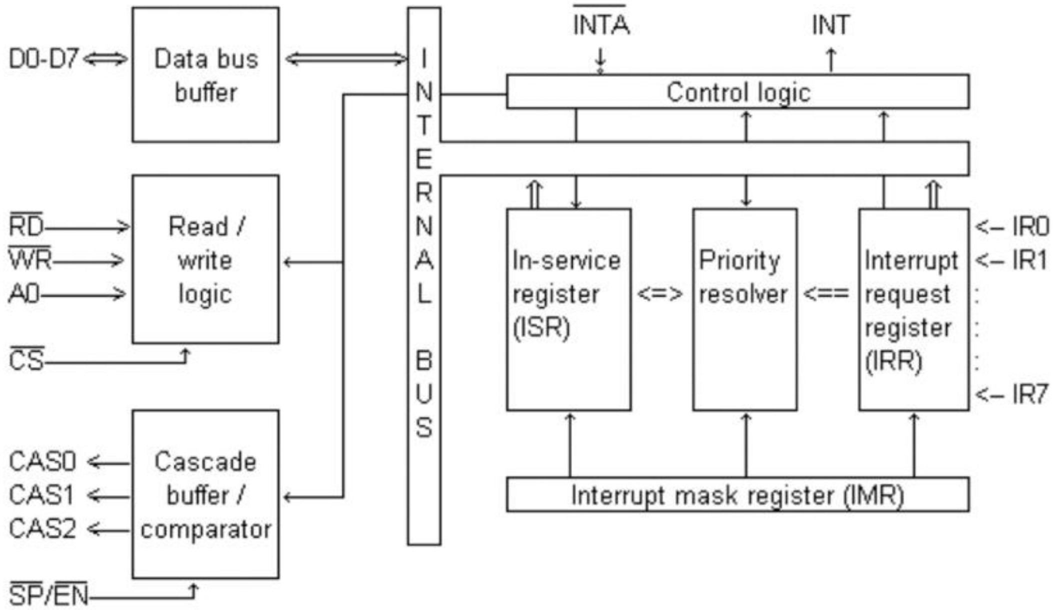
Ans.

- The Intel 8259 is a Programmable Interrupt Controller (PIC) designed for use with the 8085 and 8086 microprocessors.
- The 8259 can be used for applications that use more than five numbers of interrupts from multiple sources.

The main features of 8259 are listed below

- Manage eight levels of interrupts.
- Eight interrupts are spaced at the interval of four or eight locations.
- Resolve eight levels of priority in fully nested mode, automatic rotation mode or specific rotation mode.
- Mask each interrupt individually.
- Read the status of pending interrupt, in-service interrupt, and masked interrupt.
- Accept either the level triggered or edge triggered interrupt

8259 Internal Block Diagram



Read/Write Logic

- It is typical R/W logic.
- When address line A0 is at logic 0, the controller is selected to write a command word or read status.
- The Chip Select logic and A0 determine the port address of controller.

Control Logic

- It has two pins: INT as output and INTA as input.
- The INT is connected to INTR pin of MPU

Interrupt Registers and Priority Resolver

1. Interrupt Request Register (IRR)
2. Interrupt In-Service Register (ISR)
3. Priority Resolver
4. Interrupt Mask Register (IMR)

Interrupt Request Register (IRR) and Interrupt In-Service Register (ISR)

- Interrupt input lines are handled by two registers in cascade – IRR and ISR
- IRR is used to store all interrupt which are requesting service.
- ISR is used to store all interrupts which are being serviced.

Priority Resolver

- This logic block determines the priorities of the bit set in IRR.
- IR_0 is having highest priority, IR_7 is having lowest priority

Interrupt Mask Register

- It stores bits which mask the interrupt lines to be masked
- IMR operates on the IRR.
- Masking of high priority input will not affect the interrupt request lines.

Cascade Buffer / Comparator

This block is used to expand the number of interrupt levels by cascading two or more 8259As.

6. State the difference between the vectored and non-vectored interrupts. Explain vectored interrupts of the 8085 microprocessor.

Ans. Difference between the vectored and non-vectored interrupts

VECTORED INTERRUPT

- In vectored interrupts, the processor automatically branches to the specific address in response to an interrupt.
- In vectored interrupts, the manufacturer fixes the address of the ISR to which the program control is to be transferred.
- The TRAP, RST 7.5, RST 6.5 and RST 5.5 are vectored interrupts.
- TRAP is the only non-maskable interrupt in the 8085.

NON-VECTORED INTERRUPT

- In non-vectored interrupts the interrupted device should give the address of the interrupt service routine (ISR).
- The INTR is a non-vectored interrupt.
- Hence when a device interrupts through INTR, it has to supply the address of ISR after receiving interrupt acknowledge signal.

Interrupt	Maskable	Vectored
INTR	Yes	No
RST 5.5	Yes	Yes
RST 6.5	Yes	Yes
RST 7.5	Yes	Yes
TRAP	No	Yes

Explain vectored interrupts of the 8085 microprocessor

The vector addresses of 8085 interrupts are given below:

Software Interrupt

RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

Hardware Interrupt

RST 7.5	003CH
RST 6.5	0034H
RST 5.5	002CH
TRAP	0024H

Software Interrupt

- The software interrupts of 8085 are RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6 and RST 7.
- The software interrupts cannot be masked and they cannot be disabled.

Hardware Interrupt

- The vectored hardware interrupts of 8085 are TRAP, RST 7.5, RST 6.5, RST 5.5.
- An external device, initiates the hardware interrupts of 8O85 by placing an appropriate signal at the interrupt pin of the processor.
- The processor keeps on checking the interrupt pins at the second T -state of last machine cycle of every instruction.
- If the processor finds a valid interrupt signal and if the interrupt is unmasked and enabled, then the processor accepts the interrupt.
- The acceptance of the interrupt is acknowledged by sending an INTA signal to the interrupted device.
- The processor saves the content of PC (program Counter) in stack and then loads the vector address of the interrupt in PC. (If the interrupt is non-vectored, then the interrupting device has to supply the address of ISR when it receives INTA signal).
- It starts executing ISR in this address.
- At the end of ISR, a return instruction, RET will be placed.
- When the processor executes the RET instruction, it POP the content of top of stack to PC.
- Thus the processor control returns to main program after servicing interrupt.

7. Explain Interfacing Seven-Segment LEDs as an Output

Ans.

- Interface the 8085 Microprocessor System with seven segment display through its programmable I/O port 8255.
- Seven segment displays is often used in the digital electronic equipment to display information regarding certain process.
- I/O devices (or peripherals) such as LEDs and keyboards are essential components of the microprocessor-based or microcontroller-based systems.
- Seven-segment LEDs Often used to display BCD numbers (1 through 9) and a few alphabets.
- A group of eight LEDs physically mounted in the shape of the number eight plus a decimal point.
- Each LED is called a segment and labeled as 'a' through 'g'.

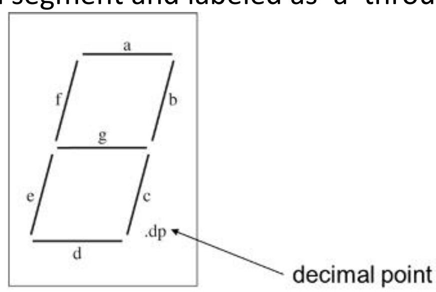
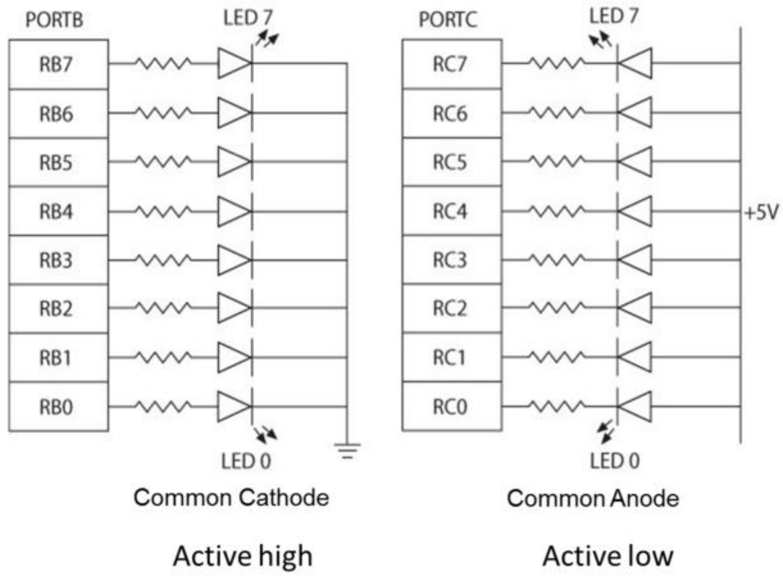


Figure: Seven Segment LED

- Commonly used output peripherals in embedded systems are LEDs, seven-segment LEDs, and LCDs; the simplest is LED

Two ways of connecting LEDs to I/O ports:

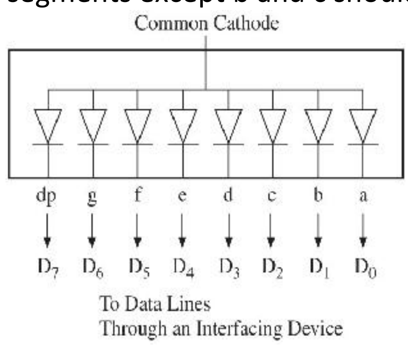
1. LED cathodes are grounded and logic 1 from the I/O port turns on the LEDs - The current is supplied by the I/O port called current sourcing.
2. LED anodes are connected to the power supply and logic 0 from the I/O port turns on the LEDs - The current is received by the chip called current sinking.



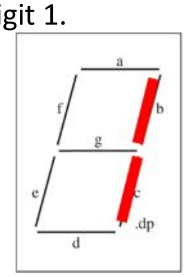
- In a common anode seven-segment LED All anodes are connected together to a power supply and cathodes are connected to data lines
- Logic 0 turns on a segment.

Example:

To display digit 1, so all segments except b and c should be off.



Byte 11111001 = F9H will display digit 1.



8. Explain I/O interfacing Methods

Ans. There are two method of interfacing memory or I/O devices with the microprocessor are as follows:

1) I/O mapped I/O

2) Memory mapped I/O

1) I/O MAPPED I/O

- In this technique, I/O device is treated as an I/O device and memory as memory. Each I/O device uses eight address lines.
- If eight address lines are used to interface to generate the address of the I/O port, then 256 Input/output devices can be interfaced with the microprocessor.
- The 8085 microprocessor has 16 bit address bus, so we can either use lower order address lines (A0 – A7) or higher order address lines(A8 – A15) to address I/O devices. We used lower order address bus & address available on A0 – A7 will be copied on the address lines A8 – A15.
- In I/O mapped I/O, the complete 64 Kbytes of memory can be used to address memory locations separately as the address space is not shared with I/O devices.
- In this interface type, the data transfer is possible between accumulator (A) and I/O devices only. Arithmetic and logical operation are not possible directly.
- As 8 bit device address used, Address decoding is simple so less hardware is required.
- The separate control signals are used to access I/O devices and memory such as IOR, IOW for I/O port and MEMR, MEMW for memory hence memory location are protected from the I/O access.

2) MEMORY MAPPED I/O

- In this technique, I/O devices are treated as memory and memory as memory, hence the address of the I/O devices are as same as that of memory i.e. 16 bit for 8085 microprocessor.
- So, the address space of the memory i.e. 64 Kbytes will be shared by the I/O devices as well as by memory. All 16 address lines i.e. A0-A15 is used to address memory locations as well as I/O devices.
- The control signals MEMR and MEMW are used to access memory devices as well as I/O devices.

Comparison of Memory-Mapped I/O and Peripheral Mapped I/O

No	Characteristics	Memory mapped I/O	I/O mapped I/O
1	Device Address	16 bit	8 Bit
2	Control signals for inputs	MEMR & MEMW	IOR & IOW
3	Instruction Available	All memory related instruction : LDA; STA; LDAX; STAX; MOV M,R; ADD M; SUB M	IN and OUT instructions only
4	Data Transfer	Between any register and I/O devices.	Between I/O device and Accumulator only.
5	Maximum Numbers of I/Os Possible	Memory Map (64K) is shared between I/Os and System memory.	I/O Mapped is independent of memory map; 256 Input and 256 output devices can be connected.
6	Execution Speed	13 T-State (LDA, STA, ..) 7 T-State (MOV M,R)	10 T-State
7	Hardware Requirement	More hardware is needed to decode 16 bit address	Less hardware is needed to decode 8 bit address
8	Other Feature	Arithmetic and logical operations are directly performed with I/O devices.	Not available

8086

1. Draw and explain 8086 Logical Block diagram

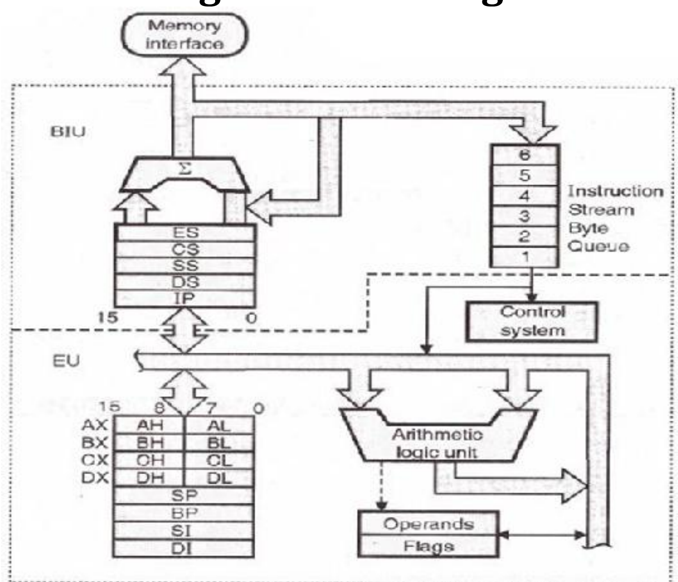


Figure: 8086 Architecture

- In 8086 CPU is divided into two independent functional parts BIU and EU.
- Dividing the work between these two units' speeds up the processing.

BIU (Bus Interface Unit)

Components of BIU

- Instruction queue
  It holds the instruction bytes of the next instruction to be executed by EU
- Segment Registers
  Four 16-bit register that provides powerful memory management mechanism
- ES (extra segment), CS (code segment), SS (stack segment), DS (data segment).
  The size of each register is 64kb.
- Instruction pointer (IP)
  Register that holds 16-bit address or offset of next code byte within code segment
- Address Generation and bus control
  Generation of 20-bit physical address

Task carried out by BIU

- Fetch instruction from memory
- Read/ Write instruction from / to the memory
- Input/ Output (I/O) of data from / to peripheral ports
- Write the data to memory.
- Address generation for memory reference
- Queuing of instruction (The instruction bytes are transferred to the instruction queue)
- Thus, BUI handles all transfer of data and address on the buses for Execution unit.
- BIU works in synchronous with machine cycles

- EU (Execution Unit)

  Components of EU

  - ALU (Arithmetic logic Unit)
    Contains 16-bit ALU, that performs add, subtract, increment, decrement, compliment, shift binary numbers, AND, OR, XOR etc.
  - CU (Control Unit)
    Directs internal operation
  - Flag Register
    16-bit flag register
    EU contains 9 active flags
  - General Purpose Registers (GPR)
    EU has 4 general purpose 16-bit register
    i.e. AX, BX, CX, DX
    each register is the combination of two 8-bit register
    AH, AL, BH, BL, CH, CL, DH, DL where 'L' means Lower byte and 'H' means higher byte.
  - Index Register
    16-bit Register is SI (source index) and DI (destination index).
    Both the register are used for string related operation and for moving block of memory from one location to the other.
  - Pointers
    16-bit Register.
    i.e. SP (stack pointer), BP (base pointer)
    BP : is used when we need to pass parameter through stack
    SP: It always points to the top of the stack. Used for sequential access of stack segment.
  - Decoder (instruction decoder)
    Translates the instruction fetched from into series of action which EU carries out

Task carried out by EU

- Decodes the instruction
- It executes instructions (executes decoded instructions)
- Tells BIU from where to fetch the instruction
- Decodes instruction (decode the fetched instruction)
- EU takes care of performing operation on the data
- EU is also known as execution heart of the processor

2. Explain 8086 Registers

Ans. The 8086 microprocessor has a total of fourteen registers that are accessible to the programmer as follows:-

General Purpose Register

AX: - Accumulator register consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX.
- AX works as an intermediate register in memory and I/O operation.
- Accumulator is used for the instruction such as MUL and DIV.

BX: - Base register consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX.
- BX register usually contains a data pointer used for based, based indexed or register indirect addressing.

CX: - Count register consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX. Count register can be used in Loop, shift/rotate instructions and as a counter in string manipulation.

DX: - Data register can be used together with AX register to execute MUL and DIV instruction.
- Data register can be used as a port number in I/O operations.

Segment Register

Types of Segment registers are as follows:-

1. Code Segment (CS): The CS register is used for addressing a memory location in the Code Segment of the memory, where the executable program is stored.
2. Data Segment (DS): The DS contains most data used by program. Data are accessed in the Data Segment by an offset address or the content of other register that holds the offset address.
3. Stack Segment (SS): SS defined the area of memory used for the stack.
4. Extra Segment (ES): ES is additional data segment that is used by some of the string to hold the destination data

Pointer Registers

The pointers IP, BP, SP usually contain offsets within the code, data and stack segments respectively.

1. Stack Pointer (SP): SP is a 16-bit register pointing to program stack in stack segment.
2. Base Pointer (BP): BP is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.
3. Instruction Pointer (IP): IP is a 16-bit register pointing to next instruction to be executed.

Index registers

The Index Registers are as follows:-

1. Source Index (SI): SI is a 16-bit register used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.
2. Destination Index (DI): DI is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data addresses in string manipulation instructions.

Flag Registers

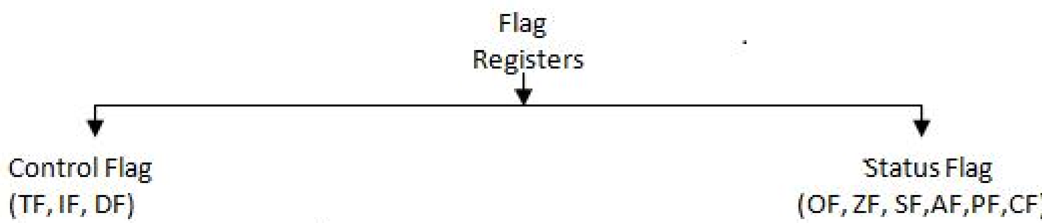
The 16-bit flag register of 8086 contains 9 active flags (six conditional & 3 control flags), other 7 flags are undefined.

3. Draw the format of a Flag register of an 8086 microprocessor.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

Figure:8086 Flag Register

- The 16-bit flag register of 8086 contains 9 active flags (six conditional & 3 control flags), other 7 flags are undefined.



- Status Flags: It indicates certain condition that arises during the execution. They are controlled by the processor.
- Control Flags: It controls certain operations of the processor. They are deliberately set/ reset by the user.

Control Flags

Control flags are set or reset deliberately to control the operations of the execution unit.

1. **Trap Flag (TF):**
   - It is used for single step control.
   - It allows user to execute one instruction of a program at a time for debugging.
   - When trap flag is set, program can be run in single step mode.
2. **Interrupt Flag (IF):**
   - It is an interrupt enable/disable flag.
   - If it is set, the mask able interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.
   - It can be set by executing instruction sit and can be cleared by executing CLI instruction.
3. **Direction Flag (DF):**
   - It is used in string operation.
   - If it is set, string bytes are accessed from higher memory address to lower memory address.
   - When it is reset, the string bytes are accessed from lower memory address to higher memory address.

Status Flag

1. **Carry Flag (CF):** This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.
2. **Auxiliary Flag (AF):** If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AF flag is set i.e. carry given by D3 bit to D4 is AF flag. This is not a general-purpose flag, it is used internally by the processor to perform Binary to BCD conversion.
3. **Parity Flag (PF):** This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity Flag is reset.
4. **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.
5. **Sign Flag (SF):** In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.
6. **Overflow Flag (OF):** It occurs when signed numbers are added or subtracted. OF=1 indicates that the result has exceeded the capacity of machine.

4. Explain Segmentation in 8086

Segment Register in 8086

Types of Segment registers are as follows:-

1. Code Segment (CS): executable program is stored in CS
2. Data Segment (DS): The DS contains most data used by program. Data are accessed in the Data Segment by an offset address or the content of other register that holds the offset address.
3. Stack Segment (SS): SS defined the area of memory used for the stack.
4. Extra Segment (ES): ES is additional data segment.

Segmentation

In Segmentation, the total memory size is divided into segments of various sizes.

What is Segment?

Segment is just an area in memory.

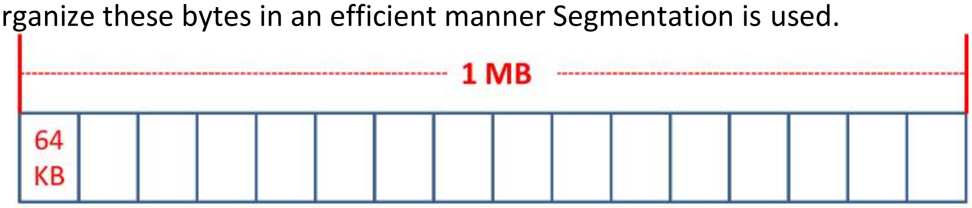
What is Segmentation?

The process of dividing memory into segments of various sizes is called Segmentation.

What is the need of segmentation in 8086?

Memory is huge collection of bytes.

In order to organize these bytes in an efficient manner Segmentation is used.



Segment = Total memory available/size of each
Segment = 1MB/64KB
= 1024KB/64KB
Total Segments = 16 segments

- Intel 8086 has 20 lines address bus.
- With 20 address lines, the memory that can be addressed is 2^{20} bytes.

2^{20} = 1,048,576 bytes
1 MB = 1111 1111 1111 1111 1111
= **FFFFF H**

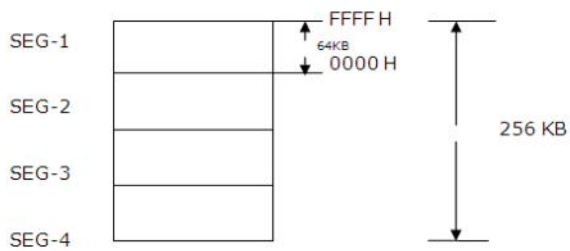


Figure: Segmentation in 8086

- 8086 memory with address ranging from 000000 H to FFFFFF H.
- Size of each Segment Register is 16-bit

2^{16} = 65535 bytes = **64K** [size of each segment]

How to calculate physical address from segment address?

- Segment Registers are used to hold the upper 16-bit of the starting address for each of the segment.
- The 16-bit address is the starting address of the segment from where the BIU is currently fetching instruction code bytes.
- The BIU always inserts zeros for the LSB of the 20-bit address for a segment, as the segment registers cannot store 20 bits, they only store the upper 16 bits.

How is a 20-bit physical address obtained if data bus is of 16-bit?

- The 20-bit address is called its Physical Address (PA).
- PA= Base Address : Offset
- Offset is the displacement of the memory location from the starting location of the segment.

E.g.

Base address DS=2222 H

Step-1: Convert DS 16-bit address to 20-bit address
- BIU appends 0H to the LSBs of the base address.
22220 H

Step-2: Retrieve offset address
- Assuming offset address = 0016 H
PA= Base Address : Offset
PA= 2222 H: 0016 H

Step-3: To calculate the effective address of the memory
Physical Address = Starting Address of Segment(20-bit) + Offset
EA = 2 2 2 2 0 H

+ 0 0 1 6 H
----
2 2 2 3 6 H

5. Explain 8086 pin diagram

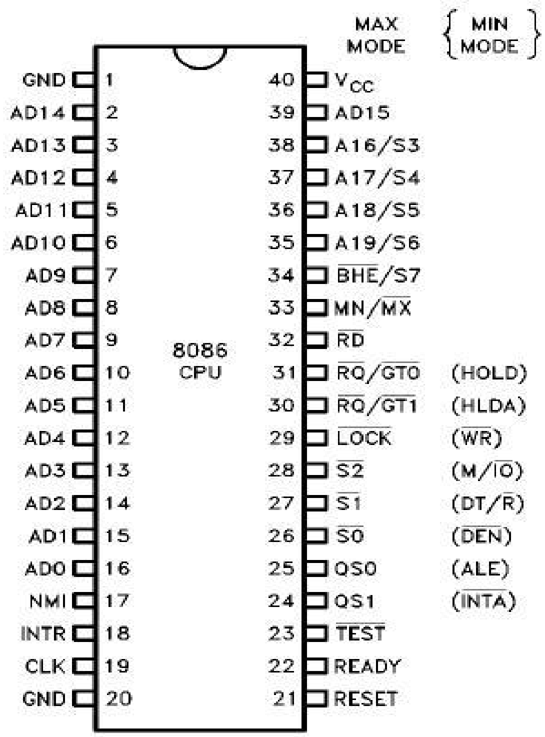


Figure:8086 Pin Diagram

Address and Data pins AD0-AD15 (bidirectional)

- These lines are multiplexed bidirectional address/data bus.
- AD0-AD7 carry lower order byte of data and AD8-AD15 carry higher order byte of data.
- When ALE=1, then Address bus gets enabled, else Data bus will get enabled.

A_{16}/S_3- A_{19}/S_6 (unidirectional)

- These lines are multiplexed and unidirectional address and status bus.
- During T_1, they carry higher order 4-bit address.
- In the remaining clock cycles, they carry status signals.
- S_5 gives the status of Interrupt Flag (IF)
- S_6 goes low, when 8086 controls the shared system bus.
- S_3 and S_4 indicates the segment register

S_4	S_3	Register
0	0	ES
0	1	SS
1	0	CS
1	1	DS

BHE/S_7

- BHE stands for Bus High Enable.
- Active low output signal.
- BHE signal is used to indicate the transfer of data over higher order data bus (D8 – D15).
- 8-bit I/O devices use this signal.
- S7 is reserved for future development.

Interrupt Related pins

NMI

- It is an active high input signal
- It is a non-mask able interrupt signal.

INTR

- It is an active high input signal
- It is an interrupt request signal.

INTA

- It is an active low output signal.
- This is an interrupt acknowledge signal.
- When microprocessor receives INTR signal, it acknowledges the interrupt by generating this signal.

Clock Related pins

CLK

Generates clock signals that synchronize the operation of processor.

RESET: Active high input signal

- When high, microprocessor enters into reset state and Terminates activity of processor
- Processor requires 4 clock cycle to reset. Thus RESET signal must be 1 for at least 4 clock cycles

READY: Active high input signal

- This is an acknowledgement signal from I/O devices or memory.
- When READY= high; it indicates that the device is ready to transfer data.
- When READY=low; microprocessor is in wait state.

Control Pins

TEST

- Active low input signal.
- It is used to test the status of math coprocessor 8087.
- The BUSY pin of 8087 is connected to this pin of 8086.
- If low, execution continues, else microprocessor is in wait state.

MN/MX

8086 works in two modes:

1. Minimum Mode [Active high i/p signal]
2. Maximum Mode [Active low i/p signal]

- If MN/MX is high, it works in minimum mode.
- If MN/MX is low, it works in maximum mode.
- Pins 24 to 31 issue two different sets of signals.
- One set of signals is issued when CPU operates in minimum mode.
- Other set of signals is issued when CPU operates in maximum mode.

Mode Multiplexed pins

S_2	S_1	S_0	Machine Language
0	0	0	Interrupt acknowledgement
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	HALT
1	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

DEN

- It is an active low output signal
- This is a Data Enable signal
- This signal is used to enable the transceiver

DT/R

- This is a Data Transmit/Receive signal.
- When high=>data is transmitted out low=>data is received in

M/IO

- This signal is issued by the microprocessor to distinguish memory access from I/O access.
- When high=> memory is accessed.

low=> I/O devices are accessed

QS_1	QS_0	Characteristics
0	0	No operation
0	1	First byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

LOCK

- It is an active low output signal.
- This signal indicates that other processors should not ask CPU (8086) to hand over the system bus.
- This pin is activated by using LOCK prefix on any instruction.

¯WR

- It is an active low output signal.
- It is used to write data in memory or output signal, depending on status of M/IO signal.

HOLD

- It is an active high input signal.
- When DMA controller needs to use address/data bus, it sends a request to the CPU through this pin.
- When microprocessor receives HOLD signal, it issues HLDA signal to the DMA controller.

HLDA

- It is an active high output signal.
- It is a Hold Acknowledge signal.
- It is issued after receiving the HOLD signal.

RQ/GT_0 and RQ/GT_1

- These are Request/Grant bi-directional pins.
- Other processors request the CPU through these lines to release the system bus.
- After receiving the request, CPU sends acknowledge signal on the same lines.
- RQ/GT_0 has higher priority than RQ/GT_1

80286

6. Describe the architecture of the 80286 with a neat block diagram.

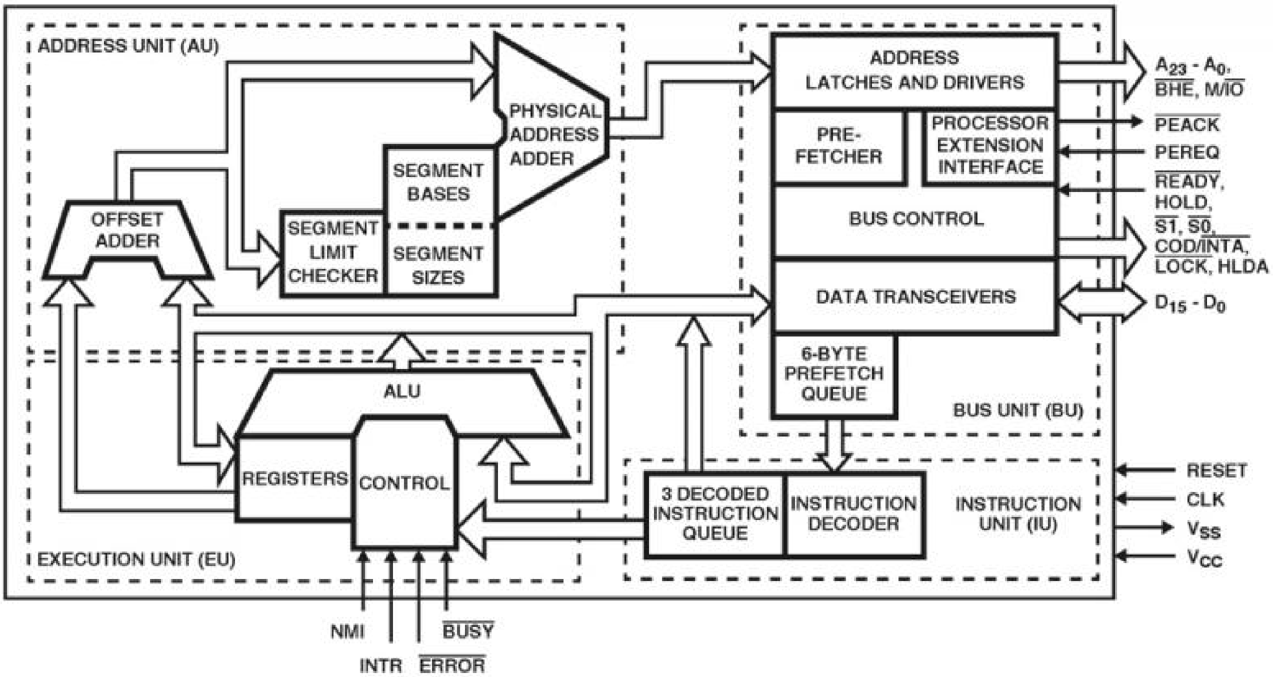


Figure: Architecture of Intel 80286

- The 80286 was designed for multi-user systems with multitasking applications, including communications and real-time process control.
- These were organized into a pipeline, significantly increasing performance.
- It was produced in a 68-pin package including PLCC (Plastic Leaded Chip Carrier), LCC (Leadless chip carrier) and PGA (Pin Grid Array) packages.
- The Intel 80286 had a 24-bit address bus and was able to address up to 16 MB of RAM, compared to 1 MB for its predecessor.

80286 Architecture contains 4 separate processing units.

1. Bus Unit (BU)
2. Instruction Unit (IU)
3. Address Unit (AU)
4. Execution Unit (EU)

Bus Unit (BU):

It has address latches, data transceivers, bus interface and circuitry, instruction pre-fetcher, processor extension interface and 6 byte instruction queue.

Functions :

- To perform all memory and I/O read and write.
- To pre-fetch the instruction bytes.

- To control the transfer of data to and from processor extension devices like 80287 math co-processor.
- Whenever BU is not using the buses for the operation, it pre-fetches the instruction bytes and put them is a 6 byte pre-fetch queue.

Instruction Unit (IU):

It has 3 decoded instruction queue and instruction decoder.

Functions :

- It fully decodes up to three prefetched instructions and holds them in a queue.
- So that EU can access them.
- It helps the processor to speed up, as pipelining of instruction is done.

Execution Unit (EU):

It includes ALU, registers and the Control unit. Registers are general purpose, index, pointer, flag register and 16 –bit Machine Status Word (MSW).

Functions :

- To sequentially execute the instructions received from the instruction unit.
- ALU result is either stored in register bank or sent over the data bus.

Address Unit (AU):

It consists of segment registers, offset address and a physical address adder.

Functions :

- Compute the physical address that will be sent out to the memory or I/O by BU.
- 80286 operate in two different modes
  1. real address mode
  2. Protected virtual address mode.
- When used in Real address mode, AU computes the address with segment base and offset like 8086. Segment register are CS, DS, ES and SS hold base address. IP, BP, SI, DI , SP hold offset.
- Maximum physical space allowed in this mode is 1MB.
- When 80286 operate in protected mode, the address unit acts as MMU.
- All 24 address lines used and can access up to 16MB of physical memory.
- If descriptor table scheme is used it can address up to 1GB of virtual memory.

7. Explain Register Organization of 80286

1. Eight 16-bit general purpose registers (AX, BX, CX, DX, SP, BP, SI, DI)
2. Four 16-bit segment registers (CS, SS, DS, ES)
3. 16-bit Instruction Pointer (IP)
4. 16-bit Flag Register Plus
5. one new 16-bit machine status word (MSW) register

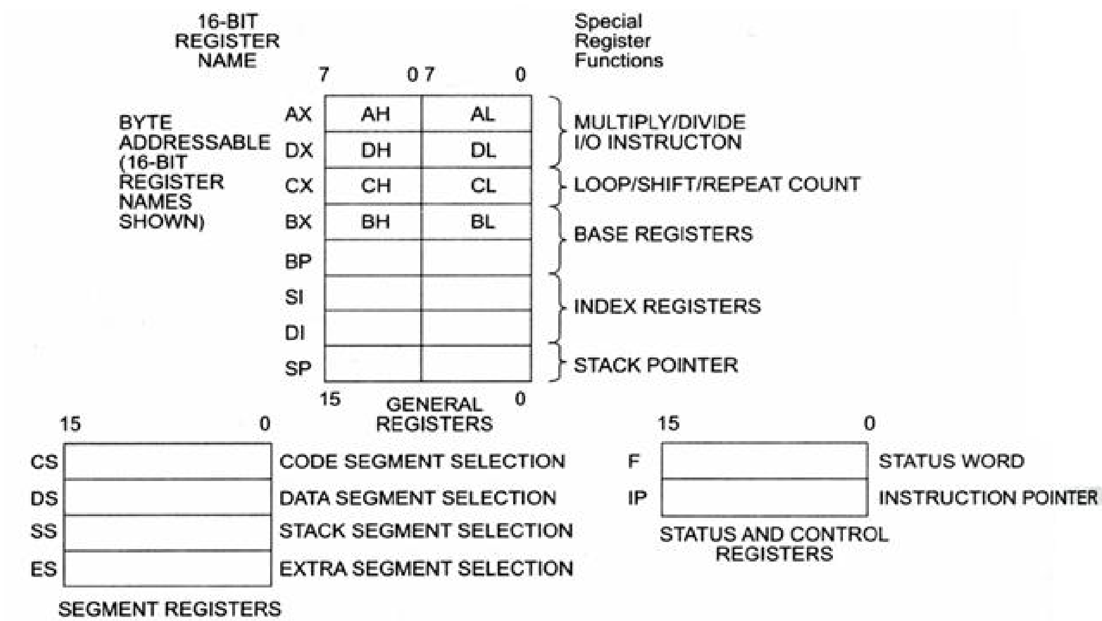


Figure: 80286 Register Set

80286 Flag Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U	NT	IOPL		OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

Figure: 80286 Flag Register

- Flag register is of 32 –bit, 15th bit undefined/reserved.
- System flags: reflect the current status of machine.
  i. **IOPL – I/O Privilege Level flag**: 2 –bits are used in protected mode. It holds the privilege level from 0 to 3. ‘0’ assigns to highest privilege whereas ‘3’ assigns to lower privilege level.
  ii. **NT: Nested Task** flag: It is used in protected mode. Bit is set when one task invokes another task.

Machine Status Word (MSW) Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
												TS	EM	MP	PE

Protected Mode Enable

PE=1; Places 80286 in protected mode

PE=0; It can be only cleared by resetting CPU

Task Switch
TS is automatically set whenever a task switch operation is performed.

Monitor processor extension
If set, this flag allows WAIT instruction to generate a processor extension not present exception

EMulate processor extension flag
If set, The EMulate coprocessor bit is set to cause all coprocessor opcodes to generate a Coprocessor Not Available fault.

Instructions used to load and store MSW:
1. LMSW instruction (Load Machine Status Word)
2. SMSW instruction (Store Machine Status Word)

8. Differentiate between the real mode and protected mode of an 80286 microprocessor.

Real Mode

- Address Unit computes the address with segment base and offset like 8086.
- Maximum physical space allowed in this mode is 1MB.
- When 80286 get reset, it always starts execution in real mode.

Task carried out in Real Mode

- Initializes IP and other registers of 80286
- Initializes the peripheral
- Enables interrupts
- Set up the descriptor table
- Prepares for entering in PVAM(Protected Virtual Address Mode)

Protected Virtual Address Mode (PVAM)

- 80286 is the 1st processor to support the concept of Virtual memory and Memory management.
- Here, the address unit acts as MMU.
- All 24 address lines are used and can access up to 16MB of physical memory.
- If descriptor table scheme is used it can address up to 1GB of virtual memory.

Task carried out in PVAM

- The complete virtual memory is mapped on to the 16Mbyte physical memory.
- If a program larger than 16Mbyte is stored on the hard disk and is to be executed by swapping sequentially as per sequence of execution.
- The huge programs are divided in smaller segments or pages arranged in appropriate sequence.

Real Address Mode	Protected Virtual Address Mode
Can only address 1 MB of system memory and act as fast 8086.	Can address till 16MB of System Memory
Doesn't Supports the concept of Virtual Memory	Supports the concept of Virtual Memory
Real mode provides no support for memory protection, multitasking, or code privilege levels.	Protected mode provides support for memory protection, multitasking, or code privilege levels.
Initially every processor is in Real Mode i.e MSW PE =0	Microprocessor will Switch to this mode by setting MSW PE-bit

Address Calculation: Real Mode

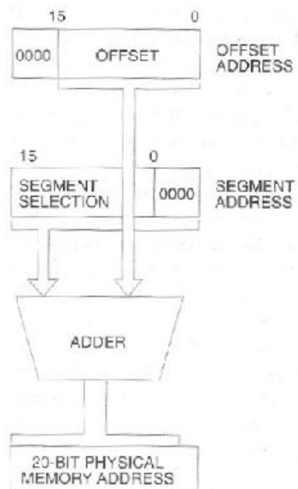


Figure: Real Mode Addressing

- Total 1MB of Memory, divided among 16-segments with each of size 64kb.
- 80286 reserves two fixed areas for
  i. System Initialization
  ii. IVT (Interrupt Vector Table)

IVT-1KB of Starting address	00000H – 003FFH
System Initialization	FFFF0H – FFFFFH

Address Calculation: PVAM

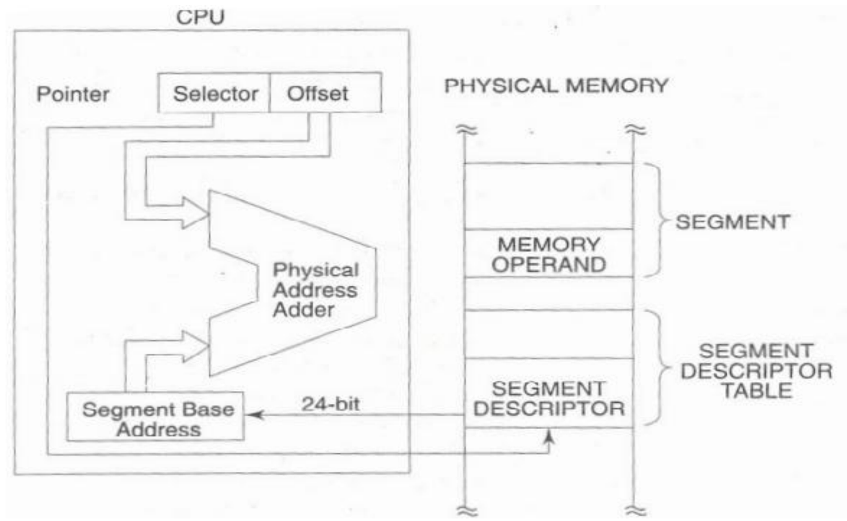


Figure: Physical Address Calculation in PVAM

- 80286 uses the 16-bit content of a segment register as a selector to address a descriptor stored in the physical memory.
- The descriptor is a block of contiguous memory locations containing information of a segment, like segment base address, segment limit, segment type, privilege level, segment availability in physical memory descriptor type and segment.
- Hardware reset is the only way to come out of protected mode

9. What is a descriptor table? What is its use? Differentiate between GDT and LDT.

Descriptor table

The descriptor is a block of contiguous memory location containing information of a segment, like

i. Segment base address
ii. Segment limit
iii. Segment type
iv. Privilege level – prevents unauthorized access
v. Segment availability in physical memory
vi. Descriptor type
vii. Segment use by another task

Use of Descriptor table

- A segment cannot be accessed, if its descriptor does not exist in either LDT or GDT.
- Set of descriptor (descriptor table) arranged in a proper sequence describes the complete program.
- The descriptor describes the location, length, and access rights of the segment of memory.
- The selector, located in the segment register, selects one of descriptors from one of two tables of descriptors.

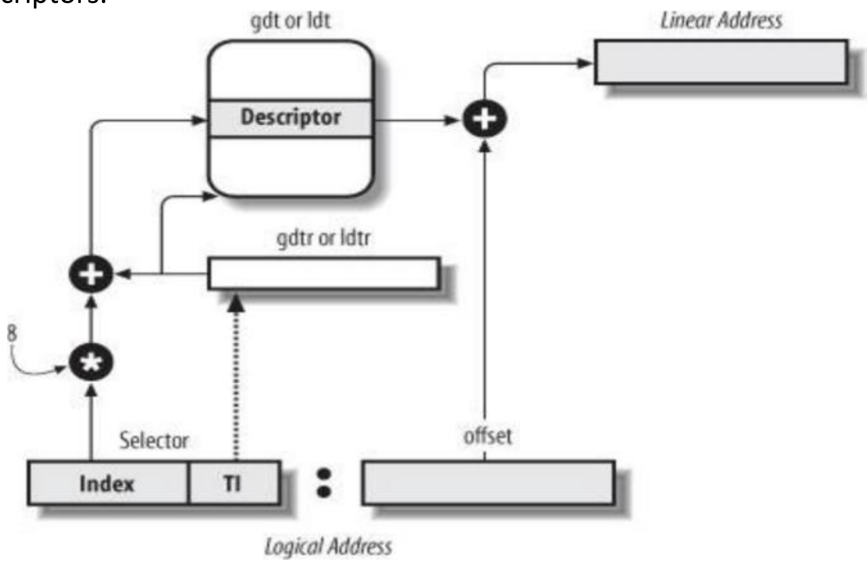


Figure: Protected Mode Addressing with Descriptor Table

Differentiate between GDT and LDT

- Each Descriptor is 8-byte long
- The GDT contains information about segments that are global in nature, that is, available to all programs and normally used most heavily by the operating system.
- The LDT contains descriptors that are application specific.
- A global descriptor might be called a system descriptor, and local descriptor an application descriptor
- The global descriptor table's base address is stored in GDTR
- The local descriptor table's base address is stored in LDTR
- GDT have only one copy in system while LDT can have many
- GDT may not change during execution while, LDT often changes when task switches
- Entry of LDT is saved in GDT.
- Entries in GDT and LDT have the same structure.
- Unlike GDT, LDT also cannot store certain privileged types of memory segments.

80386

10. Explain the Page Table and Page Directory Entry with paging mechanism in an 80386 microprocessor.

Page Directory

- The page directory contains the location of up to 1024 page translation tables, which are each four bytes long.
- Each page translation table translates a logical address into a physical address.
- The page directory is stored in the memory and accessed by the page descriptor address register (CR3).
- Control register CR3 holds the base address of the page directory, which starts at any 4K-byte boundary in the memory system.
- Each entry in the page directory translates the leftmost 10 bits of the memory address. This 10-bit portion of the linear address is used to locate different page tables for different page table entries.

Page Directory Entry

- Total Page Directory Entries are1024
- Each directory entry is of 4 byte

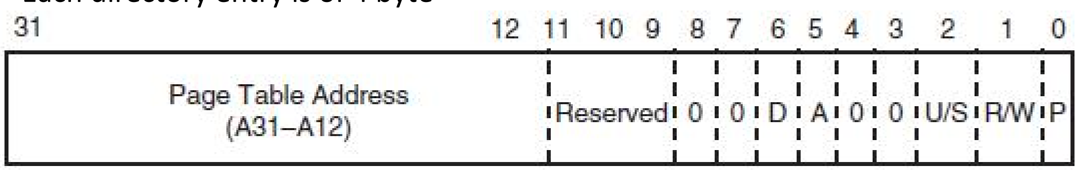


Figure: Page Directory Entry

Page Table

- The page table contains 1024 physical page addresses, accessed to translate a linear address into a physical address.

Page Table Entry

- The page table entries contain the starting address of the page and the statistical information about the page.
- Total Entries are 1024
- Each page table entry is of 4byte

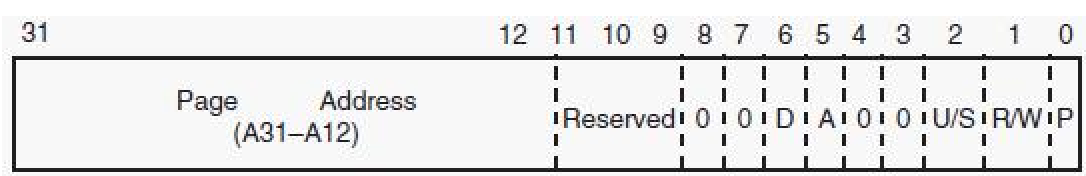


Figure: Page Table Entry

- **D-bit**: Dirty bit is undefined for page table directory entries by the 80386 microprocessor and is provided for use by the operating system.

- **A-bit:** Accessed bit is set to a logic 1 whenever the microprocessor accesses the page directory entry.
- **R/W and Read/write and user/supervisor** are both used in the protection scheme. Both bits combine to develop paging priority level protection for level 3, the lowest user level.

U/S	R/W	Access Level 3
0	0	None
0	1	None
1	0	Read-Only
1	1	Write-Only

- **P-bit**: Present bit, if logic 1 indicates that the entry can be used in address translation. If P = 0, the entry cannot be used for translation. When P = 0, the remaining bits of the entry can be used to indicate the location of the page on the disk memory system.

Page Translation Mechanism in 80386

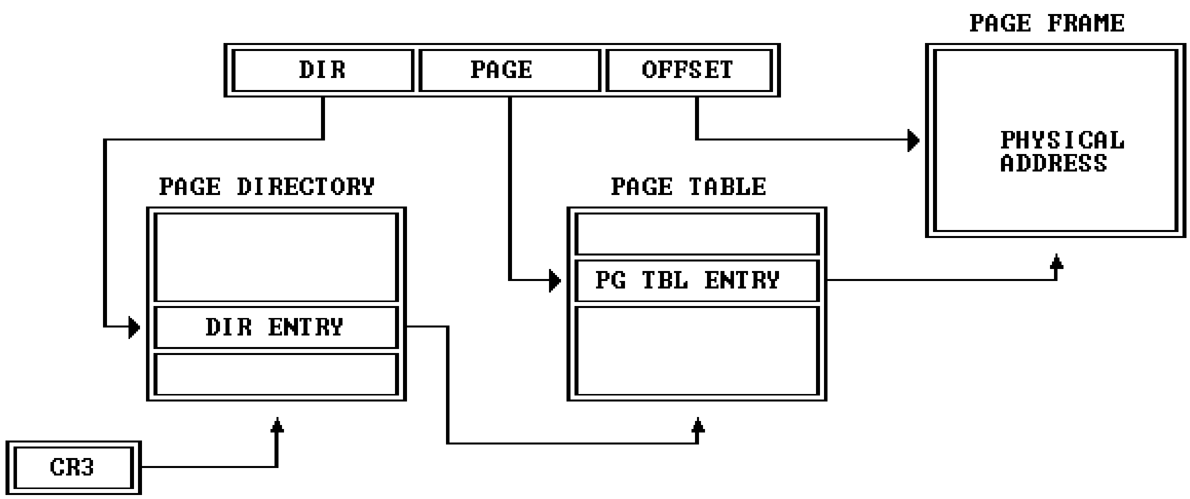


Figure: Page Translation

- A page frame is a 4K-byte unit of contiguous addresses of physical memory. Pages begin on byte boundaries and are fixed in size.
- A linear address refers indirectly to a physical address by specifying a page table, a page within that table, and an offset within that page.
- The below figure of page translation shows, how processor converts the DIR, PAGE, and OFFSET fields of a linear address into the physical address by consulting two levels of page tables.
- The addressing mechanism uses the DIR field as an index into a page directory, uses the PAGE field as an index into the page table determined by the page directory, and uses the OFFSET field to address a byte within the page determined by the page table.

- In the second phase of address transformation, the 80386 transforms a linear address into a physical address.
- This phase of address transformation implements the basic features needed for page-oriented virtual-memory systems and page-level protection.
- Page translation is in effect only when the PG bit of CR_0 is set.

11. Explain Privilege level.

- There are four types of privilege levels
- 00 - kernel level (highest privilege level)
- 01 - OS services
- 10 - OS extensions
- 11 - Applications (lowest privilege level)

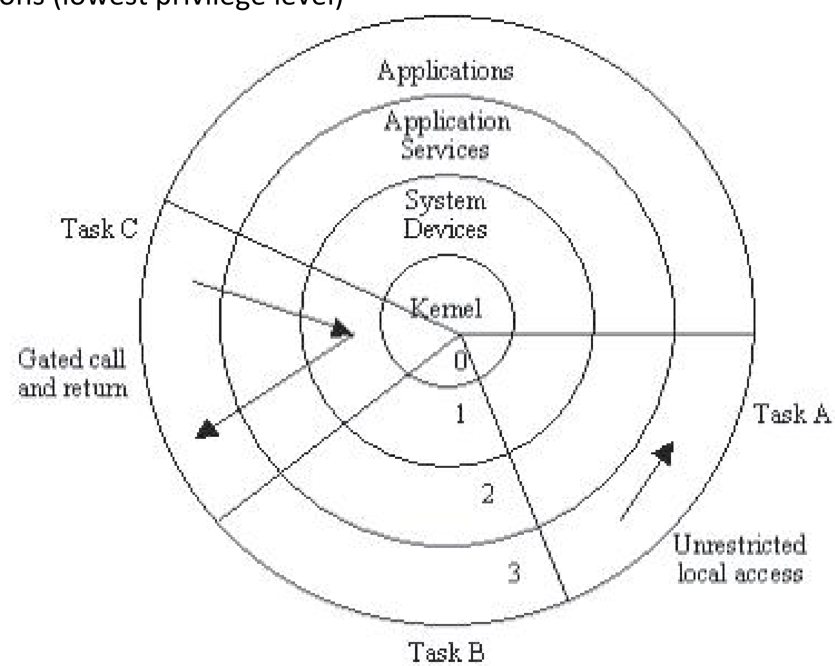


Figure: Privilege Level

- Each task assigned a privilege level, which indicates the priority or privilege of that task.
- It can only change by transferring the control, using gate descriptors, to a new segment.
- A task executing at level 0, the most privileged level, can access all the data segment defined in GDT and LDT of the task.
- A task executing at level 3, the least privileged level, will have the most limited access to data and other descriptors.
- The use of rings allows for system software to restrict tasks from accessing data.
- In most environments, the operating system and some device drivers run in ring 0 and applications run in ring 3.

12. Features of 80386

- The 80386 microprocessor is an enhanced version of the 80286 microprocessor
- Memory-management unit is enhanced to provide memory paging.
- The 80386 also includes 32-bit extended registers and a 32-bit address and data bus.These extended registers include EAX, EBX, ECX, EDX, EBP, ESP, EDI, ESI, EIP and EFLAGS.
- The 80386 has a physical memory size of 4GBytes that can be addressed as a virtual memory with up to 64TBytes.
- The 80386 is operated in the pipelined mode, it sends the address of the next instruction or memory data to the memory system prior to completing the execution of the current instruction
- This allows the memory system to begin fetching the next instruction or data before the current is completed. This increases access time.
- The instruction set of the 80386 is enhanced to include instructions that address the 32-bit extended register set.
- The 80386 memory manager is similar to the 80286, except the physical addresses generated by the MMU are 32 bits wide instead of 24-bits.
- The concept of paging is introduced in 80386
- 80386 support three operating modes:
  1. Real Mode (default)
  2. Protected Virtual Address Mode (PVAM)
  3. Virtual Mode
- The memory management section of 80386 supports virtual memory, paging and four levels of protection.
- The 80386 includes special hardware for task switching.

13. Explain the architecture of the 80386 with a neat block diagram.

- The internal architecture of the 80386 includes six functional units that operate in parallel. The parallel operation is called as pipeline processing.
- Fetching, decoding execution, memory management, and bus access for several instructions are performed simultaneously.
- The six functional units of the 80386 are
  1. Bus Interface Unit
  2. Code Pre-fetch Unit
  3. Instruction Decoder Unit
  4. Execution Unit
  5. Segmentation Unit
  6. Paging Unit

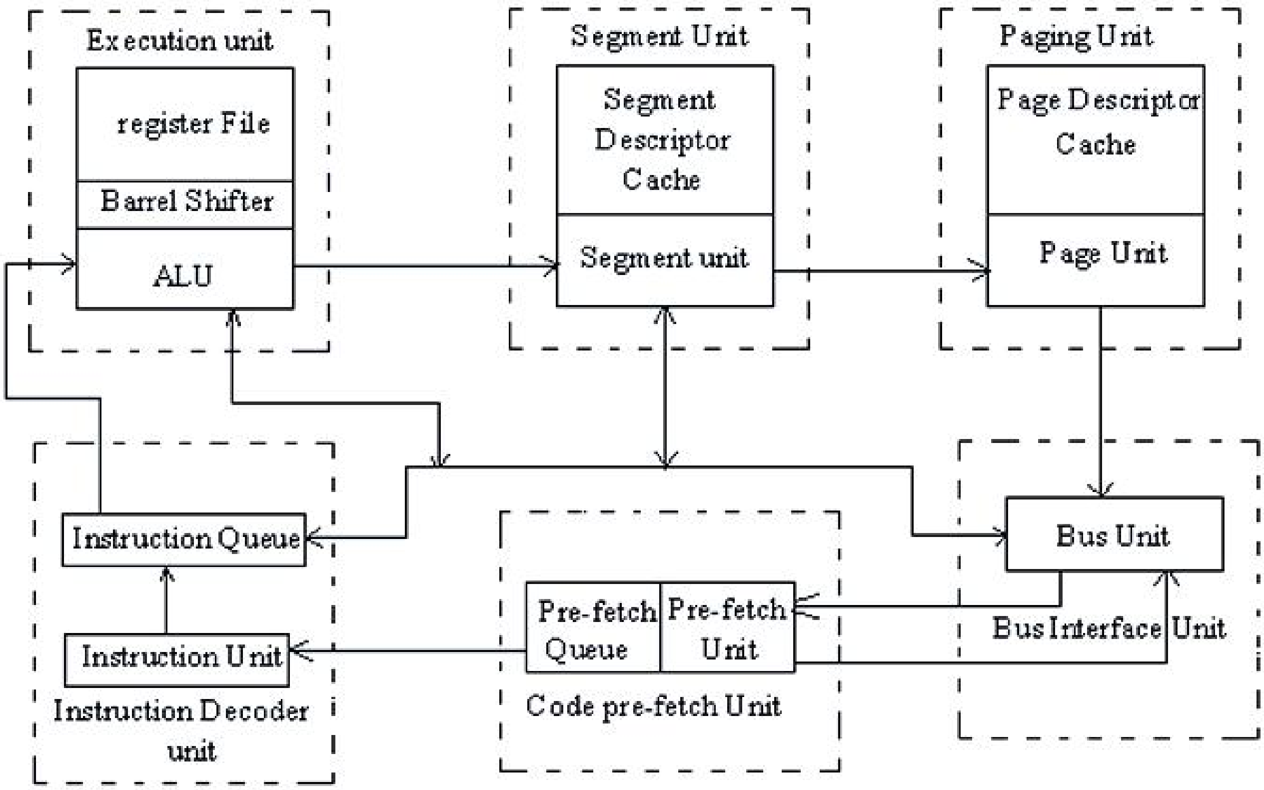


Figure: 80386 Architecture

- The Bus Interface Unit connects the 80386 with memory and I/O. Based on internal requests for fetching instructions and transferring data from the code pre-fetch unit, the 80386 generates the address, data and control signals for the current bus cycles.
- The code pre-fetch unit pre-fetches instructions when the bus interface unit is not executing the bus cycles. It then stores them in a 16-byte instruction queue for decoding by the instruction decode unit.
- The instruction decode unit translates instructions from the pre-fetch queue into micro-codes. The decoded instructions are then stored in an instruction queue (FIFO) for processing by the execution unit.
- The execution unit processes the instructions from the instruction queue. It contains a control unit, a data unit and a protection test unit.
- The control unit contains microcode and parallel hardware for fast multiply, divide and effective address calculation. The unit includes a 32-bit ALU, 8 general purpose registers and a 64-bit barrel shifter for performing multiple bit shifts in one clock. The data unit carries out data operations requested by the control unit.
- The protection test unit checks for segmentation violations under the control of microcode.
- The segmentation unit calculates and translates the logical address into linear addresses at the request of the execution unit.

- The translated linear address is sent to the paging unit. Upon enabling the paging mechanism, the 80386 translates these linear addresses into physical addresses. If paging is not enabled, the physical address is identical to the linear address and no translation is necessary.

14. Register organization of 80386

The Register organization of 80386 is as follows:

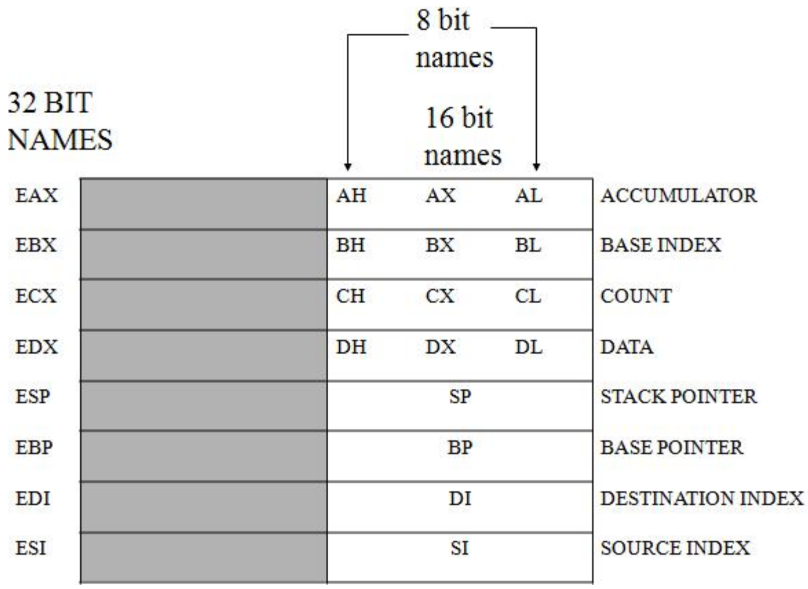


Figure:80386 General Purpose, Index and Pointer Register

General Purpose Register

- Registers EAX, EBX, ECX, EDX, EBP, EDI and ESI are regarded as general purpose or multipurpose registers.
- EAX (ACCUMULATOR): The accumulator is used for instructions such as multiplication, division and some of the adjustment instructions. In 80386 and above, the EAX register may also hold the offset address of a location in memory system.
- EBX (BASE INDEX): This can hold the offset address of a location in the memory system in all version of the microprocessor. It the 80386 and above EBX also can address memory data.
- ECX (count): This acts as a counter for various instructions.
- EDX (data): EDX is a general-purpose registers that holds a part of the result for multiplication or part of the division. In the 80386 and above this register can also address memory data.
- Pointer and Index Register
- EBP (Base Pointer): EBP points to a memory location in all version of the microprocessor for memory data transfers.
- ESP (Stack Pointer): ESP addresses an area of memory called the stack. The stack memory is a data LIFO data structure. The register is referred to as SP if used in 16 bit mode and ESP if referred to as a 32 bit register.
- EDI (Destination index): EDI often addresses string destination data for the string instruction. It also functions as either a 32-bit (EDI) or 16-bit (DI) general-purpose register.
- ESI (Source index): ESI can either be used as ESI or SI. It is often used to the address source string data for the string instructions. Like EDI ESI also functions as a general-purpose registers.

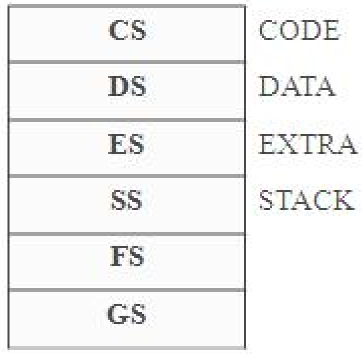


Figure:80386 Segment Register

- CS (Code): The code segment is a section of memory that holds the code used by the microprocessor. The code segment registers defines the starting address of the section of memory holding code.
- SS (Stack): The stack segment defines the area of memory used for the stack. The stack entry point is determined by the stack segment and stack pointer registers. The BP registers also addresses data within the stack segment.
- DS (Data) – The data section contains most data used by a program. Data are accessed in the data segment by an offset address of the contests of other registers that hold the offset address.
- ES (extra) – The extra segment is used to hold information about string transfer and manipulation
- FS and GS – These are supplement segment registers available in the 80386 and above microprocessors to allow two additional memory segments for access by programs.

EIP (Instruction Pointer): EIP addresses the next instruction in a section of memory defined as a code segment. This register is IP (16bit) when microprocessor operates in the real mode and EIP (32 bits) when 80386 and above operate in protected mode

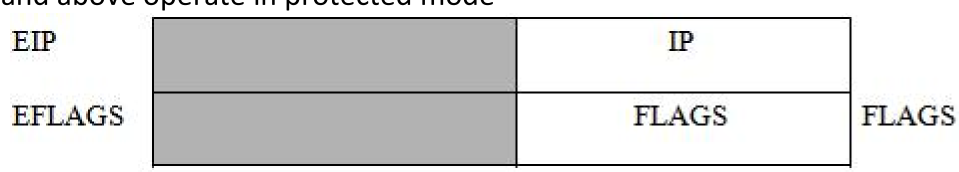


Figure:80386 Instruction Pointer and Flag Register

Flag Register:

Indicates the condition of the microprocessor and controls its operations. Flag registers are also upward compatible since the 8086-80268 have 16bit registers and the 80386 and above have EGLAF register (32 bits)

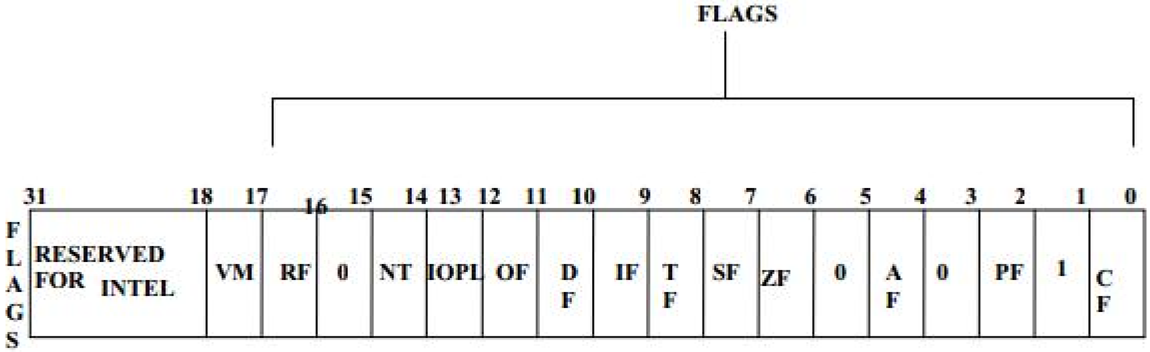
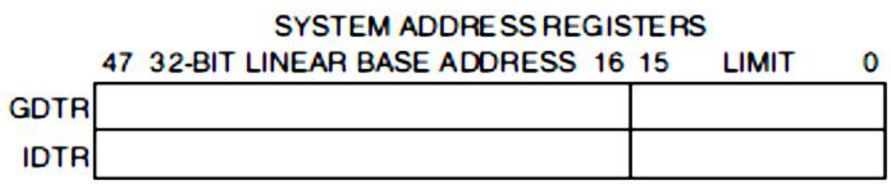
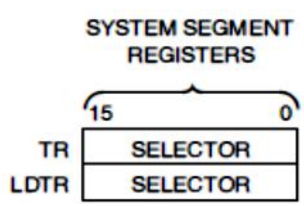


Figure: 80386 Flag Register

- IOPL (I/O Privilege level): IOPL is used in protected mode operation to select the privilege level for I/O devices. If the current privilege level is higher or more trusted than the IOPL, I/O executed without hindrance. If the IOPL is lower than the current privilege level, an interrupt occurs, causing execution to suspend. Note that an IPOL is 00 is the highest or more trusted; if IOPL is 11, it's the lowest or least trusted.
- NT (Nested Task): The nested task flag is used to indicate that the current task is nested within another task in protected mode operation. This flag is when the task I nested by software.
- RF (Resume): The resume flag is used with debugging to control the resumption of execution after the next instruction.
- VM (Virtual Mode): The VM flag bit selects virtual mode operation in a protected mode system.
- Note: All the other flag bit is having similar description as in 8086 flag register.
- System Address Register:
- Four memory management registers are used to specify the locations of data structures which control segmented memory management.
- GDTR (Global Descriptor Table Register) and IDTR (Interrupt Descriptor Table Register) be loaded with instructions which get a 6 byte data item from memory



- LDTR (Local Descriptor Table Register) and TR (Task Register) can be loaded with instructions which take a 16-bit segment selector as an operand.



Special 80386 Register

- Control Register: Four Control Register (CR0-CR3)
- Debug Register: Eight Debug Register (DR0-DR7)
- Test Register: Two Test Register (TR6-TR7)

15. Briefly explain Real, PVAM and Virtual 8086 mode of 80386 microprocessor.

Real Modes of 80386

- Default Mode
- After reset, the 80386 starts from the memory location FFFFFFF0 H under real address mode.
- In real address mode, 80386 works as a fast 8086 with 32 bit registers and data types.
- Real-address mode is in effect after a signal on the RESET pin. Even if the system is going to be used in protected mode, the start-up program will execute in real-address mode temporarily while initializing for protected mode.
- The addressing techniques, memory size, interrupt handling in this mode of 80386 are similar to the real addressing mode of 80286.
- In real address mode, the default operand size is 16 bit but 32 bit operands and addressing modes may be used with the help of override prefixed.
- Maximum physical memory = 1Mega byte (1MB)
- The only way to leave real-address mode is to switch to protected mode.

PVAM of 80386

- 32-bit address bus => access up to 232 bytes = 22. 230 B = 4 GB
- Base address => 32-bit value
- Offset =>16-bit or 32-bit value
- Linear address = base address + offset
- Linear address → physical address with paging
- In protected mode, the segment registers contain an index into a table of segment descriptors.
- Each
  segment descriptor contains the start address of the segment, to which the offset is added to generate the address.
- In addition, the segment descriptor contains memory protection information.
- This includes an offset limit and bits for write and read permission.
- This allows the processor to prevent memory accesses to certain data.
- Protected mode is accessed by placing a logic 1 into the PE bit of CR0
- This system contains one data segment descriptor and one code segment descriptor with each segment set to 4G bytes in length.
- PVAM mode support memory management, virtual memory, multitasking, protection, debugging, segmentation and paging.

Virtual Mode of 80386

- In its protected mode of operation, 80386DX provides a virtual 8086 operating environment to execute the 8086 programs.
- The real mode can also use to execute the 8086 programs along with the capabilities of 80386, like protection and a few additional instructions.
- Once the 80386 enters the protected mode from the real mode, it cannot return back to the real mode without a reset operation.
- Thus, the virtual 8086 mode of operation of 80386, offers an advantage of executing 8086 programs while in protected mode.
- The address forming mechanism in virtual 8086 mode is exactly identical with that of 8086 real mode.
- In virtual mode, 8086 can address 1Mbytes of physical memory that may be anywhere in the 4Gbytes address space of the protected mode of 80386.
- Like 80386 real mode, the addresses in virtual 8086 mode lie within 1Mbytes of memory.
- In virtual mode, the paging mechanism and protection capabilities are available at the service of the programmers.
- The 80386 supports multiprogramming, hence more than one programmer may be use the CPU at a time.
- Paging unit may not be necessarily enable in virtual mode, but may be needed to run the 8086 programs which require more than 1Mbyts of memory for memory management function.
- In virtual mode, the paging unit allows only 256 pages, each of 4Kbytes size.
- Each of the pages may be located anywhere in the maximum 4Gbytes physical memory.
- The virtual mode allows the multiprogramming of 8086 applications.
- The virtual 8086 mode executes all the programs at privilege level 3.
- Any of the other programmers may deny access to the virtual mode programs or data.
- Even in the virtual mode, all the interrupts and exceptions are handled by the protected mode interrupt handler.
- To return to the protected mode from the virtual mode, any interrupt or execution may be used.
- As a part of interrupt service routine, the VM bit may be reset to zero to pull back the 80386 into protected mode.

16. Explain features of 80486

- The 32-bit 80486 is the next evolutionary step up from the 80386.
- One of the most obvious feature included in 80486 is a built-in math coprocessor. This coprocessor is essentially the same as the 80387 processor used with a 80386, but being integrated on the chip allows it to execute math instructions about three times as fast as a 80386/387 combination.
- 80486 is an 8Kbyte code and data cache.
- To make room for the additional signals, the 80486 is packaged in a 168 pin, pin grid array package instead of the 132 pin PGA used for the 80386.
- Operates on 25MHz, 33 MHz, 50 MHz, 60 MHz, 66 MHz or 100MHz.
- It consists of parity generator/checker unit in order to implement parity detection and generation for memory reads and writes.
- Supports burst memory reads and writes to implement fast cache fills.
- Three mode of operation: real, protected and virtual 8086 mode.
- The 80486 microprocessor is a highly integrated device, containing well over 1.2 million transistors.

New feature found in the 80486 are as follows:

1. BIST (built-in self-test) that tests the microprocessor
2. 8KB Code and data cache
3. On-chip FPU(Floating Point Unit)

17. Features of Pentium Processor

It consists of all the features that 80486 has. The additional enhancements that Pentium provides are:

1. **Wider data bus width :**
   - It has 64 bit data bus and 32 bit address bus.
   - It allows 8 byte of data info to be transferred to and from memory.
   - Bus cycle pipelining has been added to allow two bus cycles to be in progress simultaneously.
2. **Improved Cache Structure:**
   - 8KB dedicated instruction cache which gives instruction to its execution units and floating point unit via dual instruction pipeline.
   - Cache is organized in a 2 way set associate cache with 32 byte line (256 lines).
   - 8KB data cache which gives data to its execution unit.
   - This allows 32 byte transfer from cache to pre-fetch buffer which is of 64 bytes.
3. **Two parallel integer execution unit :**
   - It allows the execution of two instructions to be executed simultaneously in a single processor clock.
4. **Faster floating point unit :**
   - The floating point unit has been completely redesigned over 80486.
   - Faster algorithms provide up to ten times speed – up for common operations including add, multiply etc.

5. **Branch prediction logic:**
   - The Pentium uses tech called branch prediction.
   - To implement this Pentium has two pre-fetch buffers, one to pre-fetch code in linear fashion, and one that pre-fetches code according to the Branch Target Buffer (BTB).
   - Therefore, needed code is almost pre-fetched before it is required for execution.
6. **Data Integrity and Error Detection:**
   - The Pentium have added significant data integrity and error detection capability.
   - Data parity checking is still byte-by-byte basis.
   - Address parity checking has also been added.
7. **Functional Redundancy Checking: (provide maximum error detection)**
   - Two or more Pentium Processor can participate in functional redundancy checking.
   - One processor (the master) fetching the instruction and executes the instruction in normal fashion.
   - Other processor (the checker) (connected directly to the master processor's buses)verify correctness of master processor.
   - Checker executes the instruction same as the master but doesn't drive the buses.
   - Checker samples master's output and compares the values with the internal computed values. An error signal is asserted in case if mismatch occurs.
8. **Super Scalar Architecture :**
   - Processor is capable of parallel instruction execution of multiple instructions are known as superscalar processors.
   - Pentium is capable in some cases of executing two integer of two floating point instruction simultaneously and thus support superscalar architecture.

Pentium Architecture

The term ''Pentium processor'' refers to a family of microprocessors that share a common architecture and instruction set. The first Pentium processors were introduced in 1993. It runs at a clock frequency of either 60 or 66 MHz and has 3.1 million transistors. Some of the features of Pentium architecture are

- Complex Instruction Set Computer (CISC) architecture with Reduced Instruction Set Computer (RISC) performance.
- 64-Bit Bus
- Upward code compatibility.
- Pentium processor uses Superscalar architecture and hence can issue multiple instructions per cycle.
- Multiple Instruction Issue (MII) capability.
- Pentium processor executes instructions in five stages. This staging, or pipelining, allows the processor to overlap multiple instructions so that it takes less time to execute two instructions in a row.

- The Pentium processor fetches the branch target instruction before it executes the branch instru

- The Pentium processor has two separate 8-kilobyte (KB) caches on chip, one for instructions and for data. It allows the Pentium processor to fetch data and instructions from the cache simultaneo
- When data is modified, only the data in the cache is changed. Memory data is changed only wher Pentium processor replaces the modified data in the cache with a different set of data
- The Pentium processor has been optimized to run critical instructions in fewer clock cycles than 80486 processor.

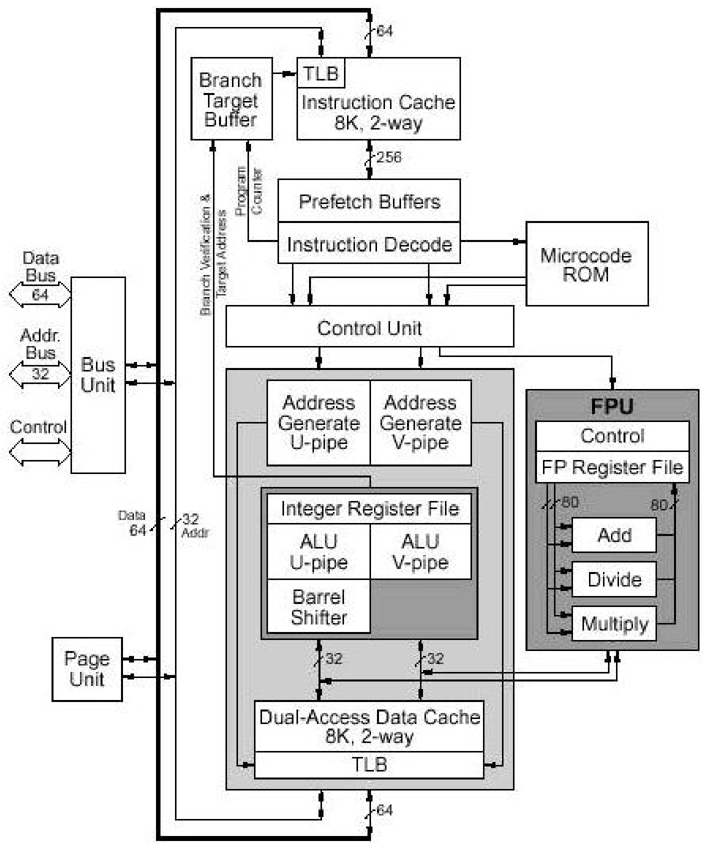


Figure: Pentium Architecture

1. Draw and explain architecture of SUN SPARC

Ans.

- **SPARC** is an acronym for **S**calable **P**rocessor **ARC**hitecture
- Engineered at Sun Microsystems in 1985
- Designed to optimize compilers and pipelined hardware implementations
- Offers fast execution rates
- SPARCs are load/store RISC processors.
- Load/store means only loads and stores access memory directly.
- RISC (Reduced Instruction Set Computer) means the architecture is simplified with a limited number of instruction formats and addressing modes.
- Simple, uniform instruction set allowing fast cycle times.
- Goal — "One instruction per cycle."(RISC)
- Up to 128 general-purpose registers
- All arithmetic operations are register-to-register
- Simplified instruction set
- Higher number of instructions with fewer transistors
- Flexible integration of cache, memory and FPUs
- 64-bit addressing and 64-bit data bus
- Increased bandwidth
- Fault tolerance
- Nine stage pipeline; can do up to 4 instructions per cycle
- On-chip 16Kb data and instruction caches with 2Mb external cache
- A large "windowed" register file — at any one instant, a program sees 8 global integer registers plus a 24-register window into a larger register file.

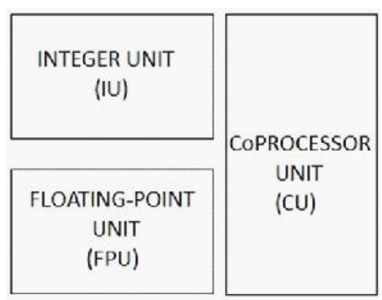


Figure: SPARC Architecture

Integer Unit:

- Contains the general purpose registers and controls the overall operation of the processor.
- Executes the integer arithmetic instructions and computes memory addresses for loads and stores.
- Maintains the program counters and controls instruction execution for the FPU.

Integer Unit: Register Window

- When a procedure is called, the register window shifts by sixteen registers, hiding the old input registers and old local registers and making the old output registers the new input registers.
- Input registers : arguments are passed to a function
- Local registers: to store any local data.
- Output registers: When calling a function, the programmer puts his argument in these registers.

Floating-point Unit (FPU)

- The FPU has
- 32 Registers (32-bit single-precision floating-point registers)
- 32 Registers (64-bit double-precision floating-point registers)
- 16 Registers (128-bit quad-precision floating-point registers)
- Floating-point load/store instructions are used to move data between the FPU and memory.
- The memory address is calculated by the IU.
- Floating-Point operate (FPop) instructions perform the floating-point arithmetic operations and comparisons.

Coprocessor Unit (CU)

- The instruction set includes support for a single, implementation-dependent coprocessor.
- The coprocessor has its own set of registers.
- Coprocessor load/store instructions are used to move data between the coprocessor registers and memory.

Block Diagram :Ultra SPARC

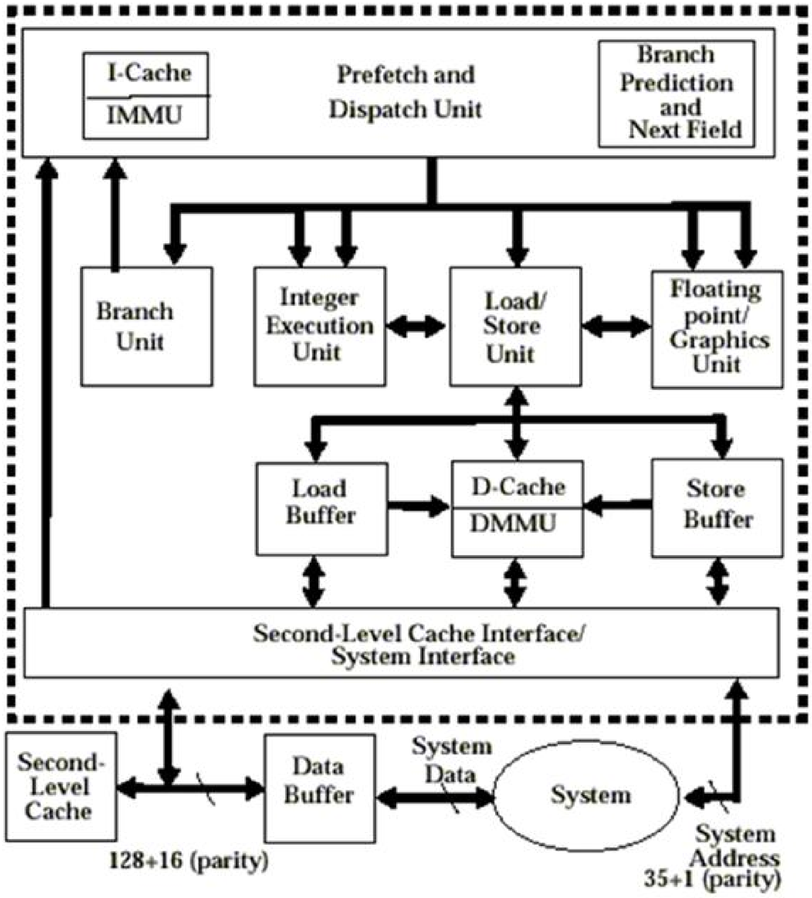


Figure: Block Diagram Ultra SPARC

2. Explain Register Window with appropriate window.

Ans.

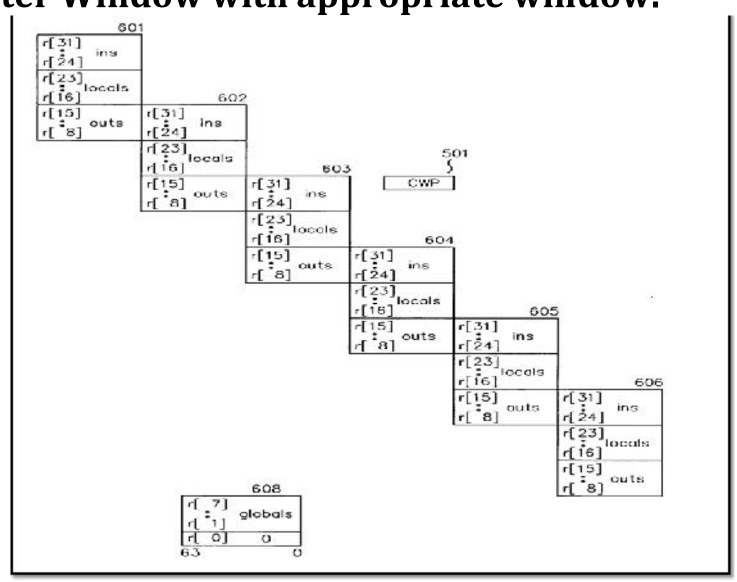


Figure: Register Window

Integer Unit: Register Window

- When a procedure is called, the register window shifts by sixteen registers, hiding the old input registers and old local registers and making the old output registers the new input registers.
- Input registers : arguments are passed to a function
- Local registers: to store any local data.
- Output registers: When calling a function, the programmer puts his argument in these registers.

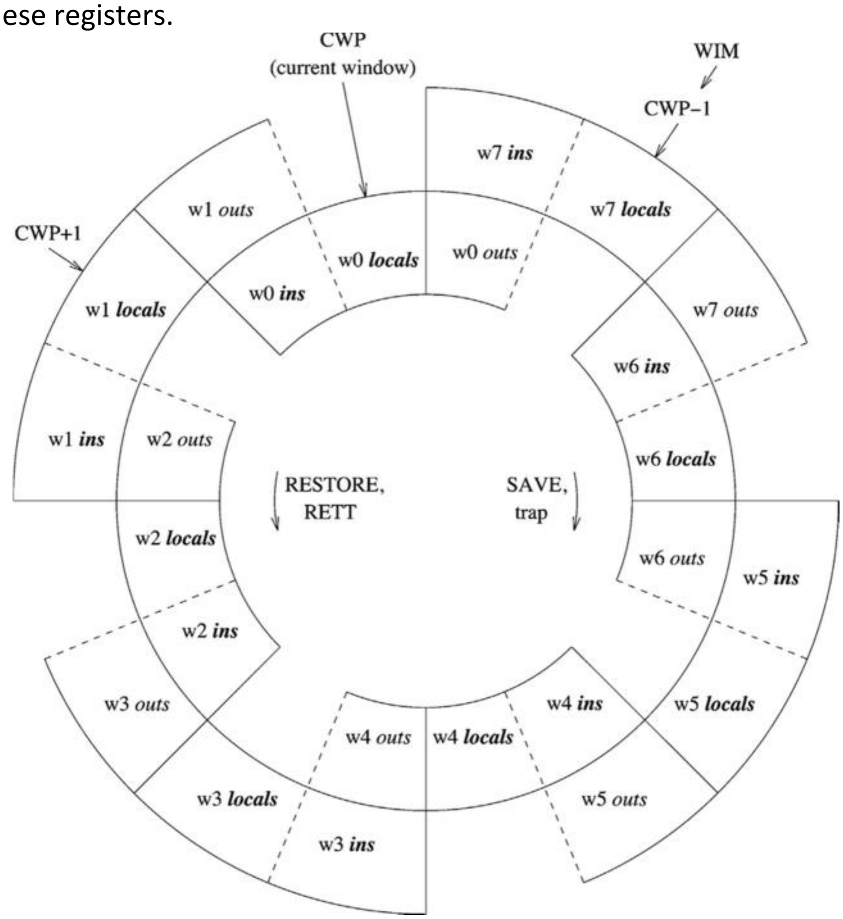


Figure: Circular Arrangement of Register Window

Advantage: Register Window

- Make very fast procedure calls as they avoid the need to save a processor's current in memory, further reducing off-chip traffic.
- Instead, the state variables are held in the current window, and the next window is opened for the new procedure.
- A refinement on this idea in that the input and output registers of adjacent windows overlap, allowing variables and parameters to be passed to the next process without physically moving data.
- The additional registers are hidden from view until you call a subroutine or other function. Where other processors would push parameters on a stack for the called

routine to pop off, SPARC processors just "rotate" the register window to give the called routine a fresh set of registers.

- The old window and the new window overlap, so that some registers are shared.

3. Data types supported by SPARC processor

Ans. **The SPARC architecture recognizes three fundamental data types:**

- Signed Integer— 8, 16, 32, and 64 bits
- Unsigned Integer— 8, 16, 32, and 64 bits
- Floating-Point — 32, 64, and 128 bits

The format widths are defined as:

- Byte — 8 bits
- Half word— 16 bits
- Word/Single word — 32 bits
- Tagged Word— 32 bits (30-bit value plus 2 tag bits)
- Double word— 64 bits
- Quad word— 128 bits

SPARC is "big-endian"- it stores multiple byte objects in memory with the most significant byte at the lowest address.

4. Explain SPARC Instruction Format

Ans. There are very few addressing modes on the SPARC, and they may be used only in certain very restricted combinations. The three main types of SPARC instructions are given below, along with the valid combinations of addressing modes. There are only a few unusual instructions which do not fall into these categories.

1. Arithmetic/Logical/Shift instructions

```
opcode reg1,reg2,reg3    !reg1 op reg2    -> reg3
opcode reg1,const13,reg3 !reg1 op const13 -> reg3
```

- All "action" instructions (add, sub, and, or, etc.) take three operands.
- The destination is always the third operand.
- The middle operand may be a 13-bit signed constant (-4096...+4095).
- Otherwise, all operands are registers.
- Examples:

```
add %L1,%L2,%L3 !%L1+%L2->%L3
```

2. Load/Store Instructions

```
opcode [reg1+reg2],reg3
opcode [reg1+const13],reg3
```

- Only load and store instructions can access memory.
- The contents of reg3 is read/written from/to the address in memory formed by adding reg1+reg2, or else reg1+const13 (a 13- bit signed constant as above).
- The operands are written in the reverse direction for store instructions, so that the destination is always last.
- One of reg1 or const13 can be omitted. The assembler will supply $g0 or 0. (This is a shorthand provided by the assembler. Both are always there in machine language.)
- Examples:

```
ld [%L1+%L2],%L3 !word at address [%L1+%L2]->%L3
```

3. Branch Instructions

```
opcode address
```

- Branch to (or otherwise use) the address given.
- There are actually 2 types of addresses but they look the same.
- Examples:

```
call printf
be   Loop
```

1. Explain ARM Architecture and Features.

Ans.

- ARM- Advanced RISC Machine
- First RISC microprocessor for commercial use
- The first ARM processor was developed in the year 1978 by Cambridge University, and the first ARM RISC processor was produced by the Acorn Group of Computers in the year 1985.
- Acorn developed a 32-bit RISC processor for its own use Acorn Archimedes
- These processors are specifically used in portable devices like digital cameras, mobile phones, home networking modules and wireless communication technologies and other embedded systems projects due to the benefits, such as
  1. Low power consumption
  2. Efficient performance

ARM Architecture: Features

- 32-bit RISC-processor core (32-bit instructions)
- A large uniform register file
- 37 pieces of 32-bit integer registers
- Pipelined (ARM7: 3 stages)
- Cached (depending on the implementation)
- 8 / 16 / 32 -bit data types
- Simple structure -> reasonably good speed / power consumption ratio
- Can have 16 coprocessors
- The ARM cortex is a complicated microcontroller within the ARM family that has ARMv7 design.
- Control over both the Arithmetic Logic Unit (ALU) and shifter in every data-processing instruction
- Maximize the use of an ALU an a shifter
- Auto-increment and auto-decrement addressing mode
- Powerful load/store multiple register instructions
- Maximize data throughputs
- Conditional execution of every instruction
- Maximize execution throughput
- Open instruction set extension through the coprocessor
- Uniform and fixed-length instruction fields, to simplify instruction decode.

2. Explain ARM Architecture in detail

Ans.

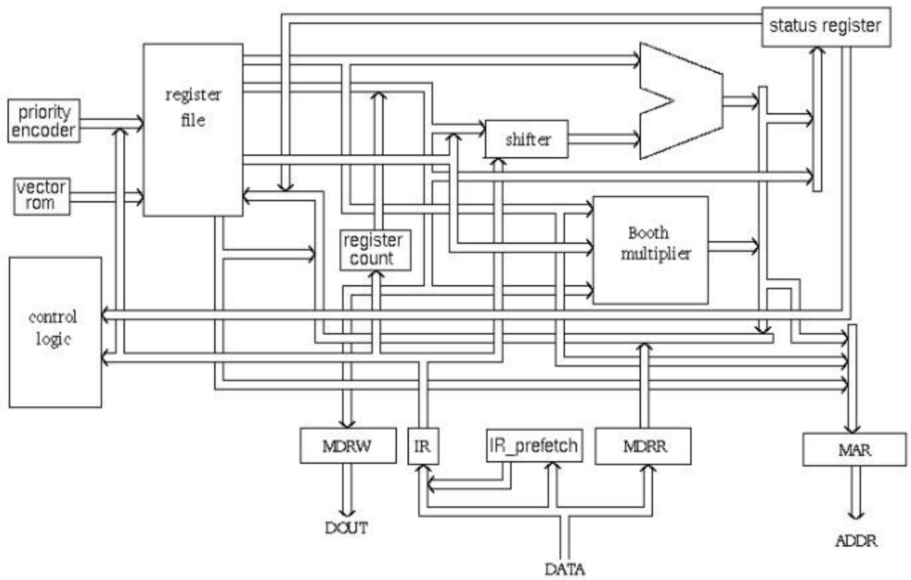


Figure: ARM Architecture

Arithmetic Logic Unit (ALU)

- The ALU has two 32-bits inputs.
- The primary comes from the register file, whereas the other comes from the shifter.
- The ALU has a 4-bit function bus that permits up to 16 Opcode to be implemented.

Booth Multiplier Factor

- The multiplier factor has 3 32-bit inputs and the inputs return from the register file.
- The multiplication starts whenever the beginning 04 input goes active. Fin of the output goes high when finishing.

Barrel Shifter

- The barrel shifter features a 32-bit input to be shifted.

Priority Encoder

- The encoder is used in the multiple load and store instruction to point which register within the register file to be loaded or kept

Control Unit

- For any microprocessor, control unit is the heart of the whole process and it is responsible for the system operation, so the control unit design is the most important part within the whole design.
- The processor timing is additionally included within the control unit.
- Signals from the control unit are connected to each component within the processor to supervise its operation.

3. ARM7 TDMI Processor

Ans.

- The ARM7TDMI core is a member of the ARM family of general-purpose 32-bit microprocessors.
- The ARM family offers high performance for very low-power consumption and gate count.
- The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles. The RISC instruction set, and related decode mechanism are much simpler than those of Complex Instruction Set Computer (CISC) designs.
- This simplicity gives:
- a high instruction throughput
- an excellent real-time interrupt response
- a small, cost-effective, processor macro cell.
- The ARM7TDMI processor uses a pipeline to increase the speed of the flow of instructions to the processor. This enables several operations to take place simultaneously, and the processing, and memory systems to operate continuously.

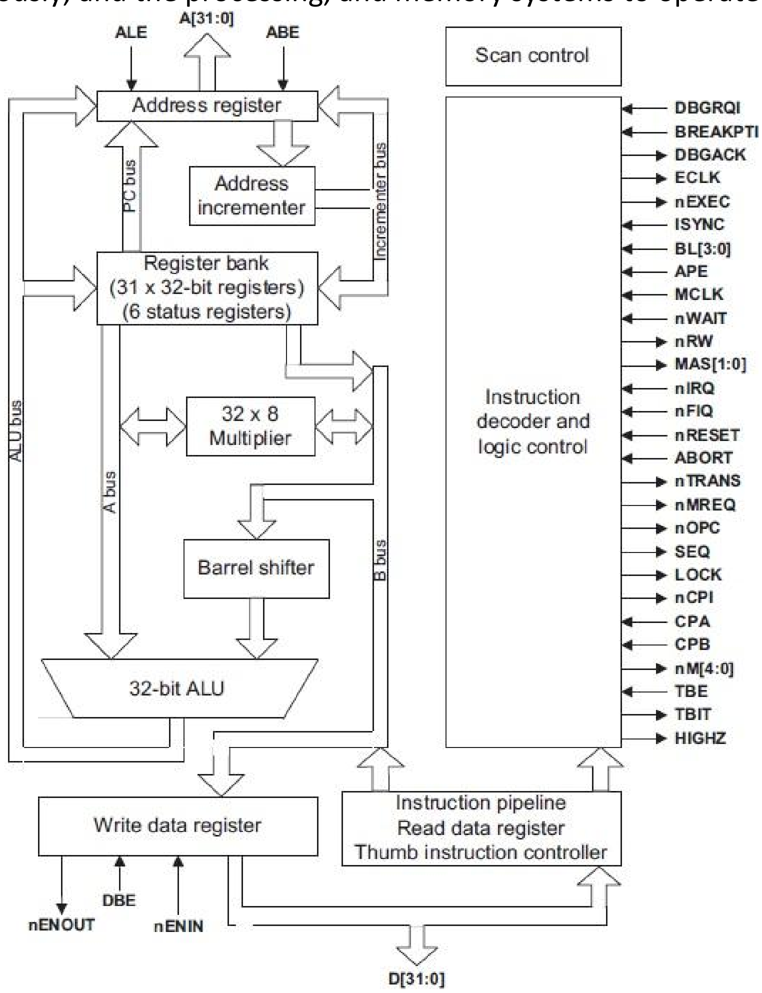


Figure: ARM7TDMI Core